WhereScape®

WhereScape RED

User Guide

Version 8.4.1.0 | July 2019

Table of Contents

Table of Contents **1**
Overview **32**
Data Flow - Enterprise Models 32
Data Flow 33
Procedural code 33
Scheduler 34
Documentation 34
WhereScape RED and Traditional ETL Tools 34
Design Introduction **35**
Design Approach 35
Objects and Windows **37**
Object Types 37
Working with Objects 41
Connections 41
Load Tables 43
Dimension Tables 49
Data Store Tables 52
EDW 3NF Tables 54
Stage Tables 55
Fact Tables/KPI Fact Tables 57
Aggregate Tables 58
OLAP Cubes 59
OLAP Dimensions 62
Procedures 64
Scripts 66
Templates 67
Indexes 68
Exports 69
Hub Tables 70
Link Tables 71
Satellite Tables 72
Source Mapping 73
Scheduler Jobs 74
Parameters 75
Object Check-Outs and Check-Ins 76
Re-create an Existing Table 80
Save these choices as the Defaults 81
Count the Rows in the Table 81
Organizing Objects 82
Object Groups 82
Projects 82
Project Groups 86
Adding Objects to Projects 87

© Copyright, WhereScape Software Ltd All rights reserved.

Adding an Object to an Independent Project (a project that is not in a group) 87
Adding an Object to a Project in a Group 87
Removing Objects from Projects 88
Using Project Object Maintenance 88
Adding an Object 90
Removing an Object 90
Adding Projects to Groups 90
Removing Projects from Groups 91
Moving Projects within Groups 92
Listing Project Memberships for Objects 92
Maintaining Project/Group Active Flag 93
Displaying/Hiding Inactive Projects/Groups 95
Building Scheduler Jobs 97
Building Scheduler Jobs from Projects or Groups 97
Building Scheduler Jobs from Object Groups 99
Building Applications 101
Building Applications from Projects or Groups 101
Building Applications from Object Groups 103
Windows and Panes 105
Builder Window 105
Toolbar 106
Scheduler Window 108
Toolbar 108
Diagram Window 109
Toolbar 110
Procedure Editor Window 111
Toolbar 112
Database Error Messages 113
Export Middle Pane Output 114
Find Function 115
Default Settings 116
Settings - Options 116
Repository Identification 117
Repository Privacy Settings 118
Changing Repository Settings 118
Username and Password Settings 118
Extract User ID Settings 118
Extract User ID Settings 119
Admin User ID Settings 119
Admin User Password Settings 119
Teradata Wallet User ID Settings 119
Teradata Wallet String Settings 119
JDBC User ID Settings 120
JDBC User Password Settings 120
Object Types 121
Object Type Availability 121

© Copyright, WhereScape Software Ltd All rights reserved.

Object Type Names 121
Object Type Ordering 122
Object Type End User Setting 123
Object Type Icon 124
To Reset An Icon 126
Reset All Icons 126
Object Type Color 126
Object Sub Types 127
Global Naming Conventions 128
Case Conversion 128
Case Conversion 129
Global Naming of Tables 129
Global Naming of Source Mappings 130
Global Naming of Indexes 131
Global Naming of Key Columns 132
Global Naming of Procedures/Scripts 133
Procedure name defaults 133
Managing Characters in Column Names 134
DSS Tables and Columns 134
dss_source_system 136
dss_fact_table 136
DSS Tables 137
Tables 138
DSS Columns 138
Columns 138
DSS Columns for Custom Targets 139
Data types for Custom Database Targets 139
Check-Out and Check-In 140
Check out 140
Retention Period 140
Code Generation 141
General 141
General 141
SQL Server Integration Services 141
Default Update Procedure Options 142
Default Update Procedure Options 142
Generated Code Statistics 144
Generated Code Statistics 144
Storage 145
Target Usage 145
Target Location 146
Target Action 147
Set Target 147
Same as Source 147
Default Target 147
Table Storage 148

© Copyright, WhereScape Software Ltd All rights reserved.

Default Optional CREATE Clause 149
ORACLE 150
Index Type 150
Foreign Key Index Type 151
Statistics 151
Metadata Versioning 152
Documentation 153
Project Documentation Field 153
Data Vault 154
Available Load Types 155
Other 156
Remove Diagram Shadow 156
User Preferences 157
Common 158
Look and Feel 158
General 158
General 158
Panes 159
Object Lists 159
Code Editor 159
Confirmation Prompts 160
Diagrams 161
Property Grids 161
Local Naming Conventions 163
General 163
Local Naming of Tables 164
Local Naming of Source Mapping 164
Local Naming of Key Columns 165
Local Naming of Indexes 166
Local Paths 166
Documentation Path 167
Backup And Restore 167
Version to Disk 167
Outputs 167
File Output 168
Middle Pane Clipboard Output 168
Other 169
Current Repository 170
Look and Feel 170
Language Options 171
Parameters 173
Managing Parameters 175
Parameter Usage 175
Global Parameters 175
Connections 176
Connection Types 177

© Copyright, WhereScape Software Ltd All rights reserved.

Database - Data Warehouse/Metadata Repository ... 177
General ... 178
Source System ... 179
Big Data Adapter Settings ... 179
Database Credentials ... 180
Other ... 180
Database ... 182
General ... 183
Source System ... 184
Database Credentials ... 184
Other ... 184
ODBC ... 186
General ... 186
ODBC ... 186
Credentials ... 187
Other ... 187
Windows ... 191
General ... 191
Windows Host ... 191
Credentials ... 192
Other ... 192
To test the connection: ... 192
UNIX ... 193
General ... 194
Unix/Linux Host ... 194
Credentials ... 195
Other ... 195
To validate the fields ... 196
To test the drag and drop functionality ... 196
Connection Failures ... 196
Closing the Connection ... 197
Hadoop ... 198
Sample Hadoop Connection screen: ... 199
General ... 199
Apache Hadoop ... 199
Credentials ... 200
Other ... 200
To test the drag and drop functionality ... 201
Closing the Connection ... 201
Hadoop using Oracle's Big Data Connectors ... 202
Hadoop for Oracle Connectors ... 202
General ... 203
Apache Hadoop ... 203
Hadoop Connectors ... 204
Oracle Loader for Hadoop ... 204
Oracle SQL Connector for HDFS ... 205

© Copyright, WhereScape Software Ltd All rights reserved.

Credentials 206
Other 206
Closing the Connection 207
Microsoft Analysis Server 2005+ 208
Microsoft Analysis Server 2005+ - OLAP Cubes 208
General 208
Online Analytical Processing (OLAP) 209
Credentials 209
Microsoft Analysis Server 2005+ - Tabular Mode 209
General 210
Online Analytical Processing (OLAP) 210
Credentials 210
Target Table Location 210
Add new Target Location 211
Browsing a Connection 211
Browser Icons 212
Connection Browse Properties 213
Changing a Connection's Properties 214
Creating a Database Link 214
Reset Meta Database Connections 216
Connection settings for BDA 217
Configuring the BDA Server 217
Big Data Adapter Settings 218
Configuring your database for use by BDA 219
Big Data Adapter Settings 219
Connection Target Settings 221
Target Connection Settings 221
Target Table Location [For target enabled licenses] 222
Connection Target Settings for Custom Database 223
Target Connection Settings 223
Target Table Location [For target enabled licenses] 225
Connection Routine Templates 226
Connection Extended Properties 227
Table Properties 228
Properties 228
Rebuilding Update Procedures 230
Generating an Update Procedure without a Template 230
Generating an Update Procedure via a Template 230
Storage 231
Table Storage Screen - SQL Server 232
Location 232
Storage 233
Other 233
Table Storage Screen - Oracle 234
Location 234
Storage 235

© Copyright, WhereScape Software Ltd All rights reserved.

Other ... 235
Dimension Tables: ... 236
Table Storage Screen - Greenplum ... 237
Location ... 237
Storage ... 238
Enable Fill Factor ... 238
Append Only ... 238
Other ... 239
Table Storage Screen - Netezza ... 239
Location ... 239
Storage ... 240
Other ... 240
Table Storage Screen - PDW ... 241
Location ... 241
Storage ... 242
Other ... 242
Table Storage Screen - Tabular ... 243
Location ... 243
Other ... 243
Processing ... 243
Table Storage Screen - Custom Database ... 244
Location ... 244
Other ... 244
Bulk Table Storage Change ... 245
Override Create DDL ... 247
Creating a table using a specific DDL statement: ... 247
Source ... 248
Documentation Fields ... 248
Documentation Fields Screen ... 249
Table Extended Properties ... 250
Notes ... 251
Distributing Tables across Multiple SQL Server Databases ... 252
Setting up Multiple SQL Databases ... 252
Step 1: Create Databases and Schema ... 252
Step 2: Create Connections and Targets ... 253
Step 3: Create New Objects ... 255
Step 4: Optional Default Target ... 257
Loading Data ... 258
Choosing the Best Load Method ... 259
Load Table Decision Tree ... 259
Load Drag and Drop ... 260
Data Type Mappings ... 261
Database Link Load ... 262
Database Link Load - Properties ... 262
Database Link Load - Source Mapping ... 264
General ... 264

© Copyright, WhereScape Software Ltd All rights reserved.

Load Table Source Mapping Screen ... 266
Remote View Extract - Oracle Databases Only ... 269
ODBC Based Load ... 270
SSIS Loader ... 271
Loading data into RED load tables using SSIS ... 272
Loading Data via SSIS from a database ... 272
Creating the SSIS Connection String ... 273
SSIS Load Table Source Screen ... 277
General ... 277
SQL Server Integration Services (SSIS) ... 279
Troubleshooting ... 282
Native ODBC Based Load ... 283
Native Loads in Oracle ... 284
If you are doing interactive Native Loads in RED into Oracle: ... 284
If you are using the Windows scheduler for Native Loads into Oracle: ... 284
Native ODBC Based Load - Source Screen ... 284
General ... 285
Native ODBC Load ... 286
Oracle and DB2 Source screen examples: ... 287
File Actions ... 288
Creating a File Action ... 289
File Action Example ... 289
Flat File Loads ... 291
Conversion ... 295
Oracle ... 295
SQL ... 296
Understanding SQL*Loader ... 296
Loading Data from Flat Files using SSIS ... 297
Flat File Load - Source Screen ... 304
File and Script based load source screens ... 308
Script Based Load ... 313
File Wild Card Load ... 314
XML File load ... 315
Supported XML Formats ... 315
Separate XML and XSD files ... 315
In line schema definition ... 318
External Load ... 318
Apache Sqoop Load ... 319
Handling Missing Source Columns ... 322
Load Table Transformations ... 324
Post-Load Procedures ... 324
Changing Load Connection and Schema ... 324
Switching Connection from ODBC to Database ... 324
Switching Connection from Database to ODBC ... 324
Dimensions ... 326
Dimensions Overview ... 326

© Copyright, WhereScape Software Ltd All rights reserved.

Dimension Keys ... 326
Business Key ... 326
Artificial Key ... 327
Building a Dimension ... 327
Drag and Drop ... 327
Dimension Type ... 327
Slowly Changing ... 328
Normal ... 328
Previous Values Dimension Type ... 329
Dimension Properties ... 330
Create and Load ... 331
Deleting and Changing columns ... 331
Adding additional columns ... 332
Manually adding previous value columns ... 333
Creating the table ... 333
Generating the Dimension Update Procedure ... 334
The zero key row ... 334
Generating a Procedure ... 334
Set Based Procedure Building types ... 334
Processing tab ... 335
Business key ... 336
Parameters ... 337
Source tab ... 339
Joining multiple source tables ... 340
Change Detection tab ... 342
Cursor procedure building types ... 343
Source Join / Lookup ... 343
Lookup ... 344
Business Key ... 345
Allow Where Clause Editing ... 346
Slowly Changing columns ... 346
Building and Compiling the Procedure ... 347
Indexes ... 347
Dimension Artificial Keys ... 348
Dimension Get Key Function ... 348
Modifying the 'Get Key' function ... 348
Adding a second 'Get Key' function for a dimension ... 349
Extending the 'Get Key' function ... 349
Dimension Initial Build ... 349
Dimension Column Properties ... 350
General ... 350
Physical Definition ... 351
Meta Definition ... 352
Code Generation ... 353
Source Details ... 354
Changing a Column Name ... 354

© Copyright, WhereScape Software Ltd All rights reserved.

Dimension Column Transformations 357
Dimension Hierarchies 357
Adding a Dimension Hierarchy 358
Copying Dimension Hierarchies from Source 359
Snowflake 359
Creating a Snowflake 359
Dimension Language Mapping 361
Staging 362
Building the Stage Table 362
Multiple Data Sources 362
Drag and Drop 363
Creating the Table 364
Generating the Staging Update Procedure 364
Generating a Procedure 364
Procedure Type 365
Oracle 366
SQL Server 367
DB2 368
Cursor Based Procedure 368
Sorted Cursor Procedure 368
Set based procedure 368
Set + Cursor Procedure 369
Set Merge Procedure 369
Set Distinct 369
Bulk Bind Procedure 369
Parameters 370
Multiple Source Tables 371
Dimension Joins 372
Slowly Changing Dimension information 373
Staging Business Key 374
Index re-creation 375
Building and Compiling the Procedure 375
Tuning the Staging Update Process 375
Removing the Update Component 376
Stage Table Column Properties 377
General 377
Physical Definition 378
Meta Definition 379
Source Details 380
Stage Table Column Transformations 381
Set Merge Procedure 382
Data Store Objects 390
Data Store Object Overview 390
Building a Data Store Object 392
Drag and Drop 392
Data Store Object Properties 392

© Copyright, WhereScape Software Ltd All rights reserved.

Create and Load 393
Deleting and Changing columns 393
Adding columns 393
Creating the table 394
Generating the Data Store Update Procedure 395
Generating a Procedure 395
Processing Tab 395
Source Tab 398
Simple Join 398
Outer Join 398
Building and Compiling the Procedure 399
Indexes 399
Data Store Artificial Keys 399
Data Store Column Properties 401
General 402
Physical Definition 402
Meta Definition 403
Code Generation 404
Source Details 404
Data Store Object Column Transformations 405
EDW 3NF Tables 406
EDW 3NF Table Overview 406
EDW 3NF Table Keys 407
Business Key 407
Artificial Key 407
Building an EDW 3NF Table 408
Drag and Drop 408
EDW 3NF Table Properties 408
Create and Load 409
Deleting and Changing columns 409
Adding columns 410
Creating the table 410
Generating the EDW 3NF Update Procedure 411
Generating a Procedure 411
Processing Tab 411
Source Tab 415
Simple Join 415
Outer Join 416
Building and Compiling the Procedure 416
Indexes 417
Converting an existing EDW 3NF Table to an EDW 3NF History Table 417
EDW 3NF Table Artificial Keys 418
EDW 3NF Table Column Properties 420
General 421
Physical Definition 421
Meta Definition 422

© Copyright, WhereScape Software Ltd All rights reserved.

Source Details 423
EDW 3NF Table Column Transformations 424
Data Vaults 425
Data Vault Functions and Features 425
Load Table Meta Data Columns 425
Data Vault Stage Table 427
Hash Key Generation Wizard 428
Hub, Link and Satellite Creation Wizard 429
Data Vault Templates 430
Data Vault Settings 431
Object Types Settings 431
Global Naming Conventions Settings 432
DSS Tables and Columns Settings 434
Data Vault 435
Table Column Properties 436
Maintain Hash Key Columns 438
Building Data Vault Objects 439
Creating Load Tables 439
Creating Data Vault Stage Tables 441
Generating Update Procedures for the Data Vault Stage Table 448
Creating the Hub, Link and Satellite Tables 451
Creating the Hub table 451
Creating the Link table 455
Creating the Satellite table 458
Generating Update Procedures for Hub, Link and Satellite Tables 463
Hub table 463
Link and Satellite Tables 465
Changing the Data Vault Hash Key Function in WhereScape RED 6.9.1.0 and above 466
Custom Objects 469
Fact Tables 470
Detail Fact Tables 470
Creating Detail Fact Tables 470
Detail Fact Columns 470
Generating the Detail Fact Update Procedure 471
Generating a Procedure 471
Define Fact Business Key Columns 471
Source Join Wizard 472
Rollup or Combined Fact Tables 472
Creating Rollup Fact Tables 473
Rollup Fact Columns 473
Generating the Rollup Fact Update Procedure 474
Generating a Procedure 474
Define Rollup date dimension and column 474
KPI Fact Tables 475
Creating the KPI Dimension 475
Creating KPI Fact Tables 476

© Copyright, WhereScape Software Ltd All rights reserved.

Adding columns 476
Setup of KPI Fact Tables 477
KPI Setup 477
KPI Dimension 477
Create the KPI Fact Table 477
Defining column types and default values 477
Create the 'Update' procedure 478
Define each KPI 479
The KPI Statement 481
Procedural Block 481
Procedure 482
Copying KPIs 482
Testing/Development 482
Generating the KPI Fact Update Procedure 483
Generating a Procedure 483
KPI parameter definition 483
Define Rollup date dimension and column 484
Select the KPI dimension and identifier 485
Snapshot Fact Tables 486
Creating Snapshot Fact Tables 486
Snapshot Fact Columns 486
Generating the Snapshot Fact Update Procedure 486
Generating a Procedure 486
Slowly Changing Fact Tables 487
Creating Slowly Changing Fact Tables 487
Partitioned Fact Tables 487
Creating a Partitioned Fact Table in SQL Server 487
Create a detail or rollup/combined fact table 487
Convert to a partitioned fact table 488
The partitioned fact table update procedure 491
Creating a Partitioned Fact Table in Oracle 491
Create a detail or rollup/combined fact table 491
Convert to a partitioned fact table 491
The partitioned fact table update procedure 495
Generating the Partitioned Fact Update Procedure 495
Generating a Procedure 495
Define Fact Business Key Columns 495
Source Join Wizard 497
Fact Table Column Properties 497
Fact Table Column Transformations 501
Fact Table Language Mapping 502
Aggregation 503
Creating an Aggregate Table 503
Change an Aggregate's Default Look Back Days 506
Creating an Aggregate Summary Table 506
Aggregate Table Column Properties 509

© Copyright, WhereScape Software Ltd All rights reserved.

General 509
Physical Definition 510
Meta Definition 510
Source Details 512
Aggregate Table Column Transformations 513
Views 514
Dimension Views 514
Creating a Dimension View: 516
View Column Re-mapping 519
Dimension View Language Mapping 520
Non-Dimension Object Views 521
Creating a Custom View 524
Creating a Custom or 'User Defined' view: 524
View Aliases 526
Analysis Services OLAP Cubes/Tabular Models 528
OLAP Cubes 528
Defining the Data Source for the OLAP Cube 529
Building the OLAP Connection String 530
OLAP Defining an OLAP Cube 532
Building a New OLAP Cube 532
Setting Cube Properties 535
OLAP Inspecting and Modifying Advanced Cube Properties 535
OLAP Creating an OLAP Cube on the Analysis Services Server 536
OLAP Cube Objects 536
OLAP Cube Properties 536
Language Mapping 540
OLAP Cube Measure Groups 541
OLAP Cube Measure Group Processing/Partitions 542
OLAP Cube Measure Group Partitions 545
OLAP Cube Measures 547
OLAP Cube Calculations 549
OLAP Cube Key Performance Indicators 551
OLAP Cube Actions 553
OLAP Cube Dimensions 555
OLAP Cube Measure Group Dimensions 557
OLAP Dimension Objects 559
OLAP Dimension Overview 559
Language Mapping 561
OLAP Dimension Attributes 562
OLAP Dimension Attribute Relationships 564
OLAP Dimension Hierarchies 565
OLAP Dimension User Defined Hierarchy Levels 566
OLAP Changing OLAP Cubes 568
OLAP Retrofitting an OLAP Object 570
Tabular Models 574
Defining the Data Source and Targets for Tabular Models 574

© Copyright, WhereScape Software Ltd All rights reserved.

Defining a Tabular Model 578
Tabular Model Measures 580
Tabular Model Calculated Columns 582
Exporting Data 584
Building an Export Object 584
File Attributes 587
Export File Definition 588
Trigger File 589
File Actions 590
File Attributes - SSIS Exports 591
Export File Definition 591
SQL Server Integration Services (SSIS) 592
Export Column Properties 592
General 592
Physical Definition 593
Meta Definition 593
Source Details 593
Script Based Exports 595
Transformations 596
Column Transformations 596
Column Transformation Properties 597
Load Table Column Transformations 598
Database Link During Load Transformations 599
File During Load Transformations 601
After Load Transformations 601
Database Functions 602
Re-usable Transformations 602
Creating a New Re-usable Transformation 602
Specify the Name of the Transformation 603
Enter Re-usable Transformation Metadata 604
Define the Transformation Model 605
Completed Re-usable Transformation 606
Changing a Re-usable Transformation 606
Applying Changes to Dependant Transformations 607
Using Re-usable Transformations 608
Procedures and Scripts 609
Procedures 609
Scripts 609
Procedure Generation 609
Wrapper procedures 610
Procedure Placeholders 612
Procedure Editing 614
Procedure Loading and Saving 618
Procedure Comparisons 619
Procedure Compilation 620
Procedure Running 621

© Copyright, WhereScape Software Ltd All rights reserved.

Procedure Syntax 621
Procedure Properties 622
Procedure Migration 625
Connection Target Settings 625
Procedure Properties 626
Procedure Placeholders 626
Moving Existing Procedures 627
Non-modified 627
Modified 627
Applying the procedure and table mark-ups in the procedure 627
Moving the procedure to the new database 627
Script Generation 628
Script Generation (Windows/SQL Server) 629
Windows PowerShell Scripts 632
Via the WhereScape RED Script Editor 633
Via the Stub Template 635
Script Generation (UNIX/Oracle) 637
Script Generation (Windows/Oracle) 642
Script Editing 646
Script Testing 647
UNIX Scripts 647
Script Syntax 647
Script Environment Variables 648
Calling a Batch File from a Script 653
Scheduling Scripts 656
Manually Created Scripts 658
Templates 659
Template Properties 660
Properties Screen Banner 662
Pebble Templates 662
Operation Types 662
Create DDL 662
Custom Procedure 662
Export Script 662
Index Create DDL 662
Index Drop DDL 662
Load Script 663
Table Alter DDL 663
Update Procedure 663
Types 663
ActualTable 663
AdditionalDatum 663
ChangeType 663
Column 664
ColumnAddition 665
ColumnAdditionalInformationChange 665

© Copyright, WhereScape Software Ltd All rights reserved.

ColumnDataTypeChange 665
ColumnDeletion 665
ColumnJoin 665
ColumnNameChange 666
DateBasedPartitioningInfo 666
EnumValue 666
Env 666
ExportInfo 666
ExtendedPropertyDefinition 667
ExtendedPropertyScope 667
FK 667
FileInfo 667
HiveNativeDelimitedRowFormat 668
HiveNativeFileFormat 668
HiveNativeSerdeRowFormat 668
HiveNativeStorage 668
HiveNonNativeStorage 668
HiveStorage 668
LoadInfo 669
Options 670
PDWLoadOptions 670
Relation 670
Relationship 671
SSISLoadOptions 671
SourceJoinDetails 671
SourceTableInfo 671
SqoopLoadOptions 672
Storage 672
TPTLoadOptions 672
Table 673
TableAdditionalInformationChange 674
TableCreation 674
UpdateProcSettings 674
VanillaSettings 675
ViewInfo 675
Template Editor 676
Evaluating an API Outline Template 677
Pebble Syntax 681
Controls 681
Pebble Template Tags 682
autoescape 682
block 682
extends 682
filter 682
for 682
for else 683

© Copyright, WhereScape Software Ltd All rights reserved.

if ... 683
if else ... 683
if elseif ... 683
import ... 683
include ... 684
include with ... 684
macro ... 684
parallel ... 684
set ... 684
verbatim ... 684
br ... 685
counter ... 685
declare ... 685
error ... 686
indent ... 686
from ... 686
from else ... 686
list add ... 687
list adds ... 687
list remove ... 687
list removes ... 688
map put ... 688
map remove ... 688
placebottom ... 688
get ... 688
WhereScape RED Specific Tag ... 689
fetch ... 689
Pebble Template Functions ... 689
block ... 689
max ... 689
min ... 689
parent ... 690
range ... 690
range with step ... 690
bool ... 690
codepoint ... 690
combine lists ... 691
double ... 691
int ... 691
replace start ... 692
replace end ... 692
str is equals ... 692
starts with ignore case ... 692
ends with ignore case ... 693
Pebble Template Filters ... 693
abbreviate ... 693

© Copyright, WhereScape Software Ltd All rights reserved.

abs ... 693
capitalize ... 693
date ... 694
default ... 694
escape ... 694
first ... 695
join ... 695
last ... 695
length ... 696
lower ... 696
merge ... 696
numberformat ... 696
raw ... 697
replace ... 697
rsort ... 697
slice ... 697
sort ... 698
title ... 698
trim ... 698
upper ... 698
urlencode ... 698
lines ... 699
Pebble Template Tests ... 699
defined ... 699
empty ... 699
even ... 699
iterable ... 700
map ... 700
null ... 700
odd ... 700
boolean ... 700
instanceof ... 701
number ... 701
string ... 701
Pebble Template Variables ... 702
StringUtil ... 702
Template Usage ... 709
Script Templates for Custom Database Table Objects ... 710
Windows PowerShell Templates ... 714
PowerShell Stub Template (wsl_common_powershellscript_stub) ... 714
Scheduler ... 715
Scheduler Window ... 715
Auto Menu ... 718
Logs Menu ... 718
Tools Menu ... 719
Select Job Report Fields ... 720

© Copyright, WhereScape Software Ltd All rights reserved.

Scheduler States 721
Scheduling a Job 723
Working with Jobs 728
Job Tasks 729
Creating a Job 731
Adding a task 733
Setting the action on a task 733
Setting the state of a task 737
Creating dependencies between tasks 738
Ordering or Grouping the tasks 739
Editing a Job 740
Editing Tasks in a Job 743
Adding a task 744
Setting the action on a task 744
Creating dependencies between tasks 748
Ordering or Grouping the tasks 749
Editing Task Dependencies 750
Show Dependencies Diagram 754
Inserting a Copy of a Job 755
Deleting a Job 756
Deleting Job Logs 757
Starting a Job 759
Halting a Job 759
Aborting a Job 760
Effects of Aborting a Job 760
Windows Scheduler 760
Unix Scheduler 761
Restarting a Job 762
Creating an Application from a Job 765
Monitoring the Database and Jobs 768
Database Monitoring 768
Job Monitoring 769
Monitor Days 771
Nominal Start Time 771
Earliest Possible Start 771
Periodicity of Job 771
Notification Order 771
Must Start Within (Notification) 771
Must Finish In (Notification) 772
Run Warning Count (Notification) 772
Run Error Count (Notification) 772
Finish Warning Count (Notification) 772
Finish Error Count (Notification) 772
Finish Success (Notification) 772
Checkpoint 1 (Notification) 772
Checkpoint 2 (Notification) 772

© Copyright, WhereScape Software Ltd All rights reserved.

Checkpoint 3 (Notification) ... 772
Stand Alone Scheduler Maintenance ... 773
Scheduler maintenance logon ... 773
Scheduler maintenance grants ... 773
SQL to return Scheduler Status ... 775
Reset Columns in Job and Task View ... 775
Scheduling a RED Host Script from 3D ... 776
Stopping a Linux/UNIX Scheduler from within RED ... 783
Indexes ... 785
Index Definition ... 785
Properties ... 786
Extended Properties ... 790
Documentation and Diagrams ... 791
Creating Documentation ... 791
Create documentation ... 791
Creating a header ... 792
Adding Links ... 792
Adding glossary elements ... 793
Batch Documentation Creation ... 793
Parameter Descriptions ... 793
Reading the Documentation ... 794
Diagrams ... 795
Types of Diagrams ... 795
Schema Diagram ... 796
Source Diagram ... 798
Creating a Job from a Source Diagram ... 799
Joins Diagram ... 802
Links Diagram ... 803
Impact Diagram ... 804
Dependency Diagram ... 805
Editing a Job's Dependencies from a Job Dependency diagram ... 805
Working with Diagrams ... 806
Diagram Display ... 806
Diagram Save ... 806
Diagram Load ... 807
Diagram Print ... 807
Diagram Refresh ... 808
Creating a Job from a Diagram ... 809
Creating an Application from a Diagram ... 811
Creating a Project from a Diagram ... 812
Query Tool Access ... 814
Dimension tables and columns ... 814
Fact tables and columns ... 814
Fact - Dimension Joins ... 814
Reports ... 815
Dimension-Fact Matrix ... 815

© Copyright, WhereScape Software Ltd All rights reserved.

OLAP Dimension-Cube Matrix 815
Dimension Views of a Specified Dimension 816
Column Reports 817
Columns without Comments 817
All Column Transformations 818
Sending Results to Microsoft Excel 819
Re-Usable Column Transformations 819
Column Track-Back 819
Column Track-Forward 821
Table Reports 823
Tables without Comments 823
Load Tables by Connection 824
Export Objects by Connection 825
Records that failed a Dimension Join 825
External Source Tables/Files 826
Procedure Reports 827
Modified Procedures 827
Custom Procedures 828
Index Reports 829
Modified Indexes 829
Object Reports 829
Objects-Project Matrix 829
Modified Objects (excluding indexes) 830
Objects Checked-out 831
Loaded or Imported Objects 832
Objects with Extended Properties 833
Job Reports 834
Objects by Scheduler Job 834
Jobs with an Object 835
Tasks of a Job 836
Operational Reports 837
Object Performance History 837
Job Performance History 838
Task Performance History 839
Fragmentation 840
Validate 841
Validate Meta-data 841
Sending Results to Microsoft Excel 841
Validate Workflow Data 841
Sending Results to Microsoft Excel 841
Validate Table Create Status 841
Sync Column order with database 841
Sending Results to Microsoft Excel 841
Validate Load Table Status 841
Sending Results to Microsoft Excel 841
Validate Index Status 842

© Copyright, WhereScape Software Ltd All rights reserved.

Sending Results to Microsoft Excel 842
Validate Procedure Status 842
Sending Results to Microsoft Excel 842
List Metadata Tables not in the Database 842
Objects Included 842
Parameters 842
Results 842
Sending Results to Microsoft Excel 843
List Database Tables not in the Meta-data 843
Objects Included 843
Parameters 843
Results 843
Sending Results to Microsoft Excel 843
List Tables with no related Procedures or Scripts 844
Objects Included 844
Parameters 844
Results 844
Sending Results to Microsoft Excel 844
List Procedures not related to a Table 845
Objects Included 845
Parameters 845
Results 845
Sending Results to Microsoft Excel 845
Compare Meta-data Repository to Another 846
Objects Included 846
Parameters 846
Results 847
Checking result details: 847
Sending Results to Microsoft Excel 847
Compare Meta-data Indexes to Database 847
Objects Included 847
Parameters 847
Results 848
Sending Results to Microsoft Excel 848
List Duplicate Business Key Columns 848
Objects Included 848
Parameters 848
Results 848
Sending Results to Microsoft Excel 849
Query Data Warehouse Objects 849
Objects Included 849
Parameters 849
Results 850
Sending Results to Microsoft Excel 850
Promoting Between Environments 851
Applications 851

© Copyright, WhereScape Software Ltd All rights reserved.

Application Creation 851
Application 853
Objects to Add/Replace 855
Objects to Delete 856
Application Loading 857
Creating and Loading Applications from the Command Line 857
The process typically involves the following steps: 858
Importing Object Metadata 861
Importing Language Files 863
Data Warehouse Testing 863
Backing Up and Restoring Metadata 867
Backup using DB Routines 867
UNIX Backups 867
Windows Backups 867
Oracle Windows Backups 867
SQL Server Windows Backups 868
DB2 Windows Backups 868
Restoring DB Backups 869
Restoring metadata 869
Oracle Windows Restore 869
Oracle UNIX restore 870
SQL Server Windows Restore 870
DB2 Windows Restore 871
Unloading Metadata 871
Performing an Unload 872
Loading an Unload 874
Performing a Load 874
Altering Metadata 877
Validating Tables 877
Using outdated metadata in drag and drop 878
Validating Source (Load) Tables 878
Validating Procedures 879
Altering Tables 880
Validating Indexes 881
Recompiling Procedures 881
Re-validate Procedures (Oracle only) 882
Callable Routines 883
Introduction to Callable Routines 883
Callable Routines API 883
Callable Routines per RDBMS 884
Callable Routines Names Qualifier 884
Callable Routines Common Input 885
Callable Routines Invocation 885
Alternative Invocation Methods 886
Invocation using Named Parameters 886
Invocation via ODBC 887

© Copyright, WhereScape Software Ltd All rights reserved.

Invocation via the Command-Line ... 888
Ws_Api_Glossary ... 888
SQL Server ... 889
Teradata ... 889
Oracle ... 889
DB2 ... 890
Ws_Connect_Replace ... 890
SQL Server ... 891
Teradata ... 892
Oracle ... 893
DB2 ... 893
Ws_Job_Abort ... 894
SQL Server ... 895
Teradata ... 895
Oracle ... 895
DB2 ... 896
Ws_Job_Clear_Archive ... 896
SQL Server ... 897
Teradata ... 898
Oracle ... 899
DB2 ... 899
Ws_Job_Clear_Logs ... 901
SQL Server ... 901
Teradata ... 902
Oracle ... 903
DB2 ... 904
Ws_Job_Clear_Logs_By_Date ... 905
SQL Server ... 905
Teradata ... 906
Oracle ... 907
DB2 ... 908
Ws_Job_Create ... 909
SQL Server ... 910
Teradata ... 912
Oracle ... 913
DB2 ... 914
Ws_Job_CreateWait ... 915
SQL Server ... 916
Teradata ... 918
Oracle ... 919
DB2 ... 920
Ws_Job_Dependency ... 921
SQL Server ... 922
Teradata ... 923
Oracle ... 924
DB2 ... 925

© Copyright, WhereScape Software Ltd All rights reserved.

Ws_Job_Release 926
SQL Server 926
Teradata 927
Oracle 928
DB2 929
Ws_Job_Restart 929
SQL Server 930
Teradata 931
Oracle 932
DB2 932
Ws_Job_Schedule 933
SQL Server 934
Teradata 935
Oracle 935
DB2 936
Ws_Job_Status 937
SQL Server 939
Teradata 940
Oracle 941
DB2 942
Ws_Load_Change 943
SQL Server 944
Teradata 944
Oracle 945
DB2 946
Ws_Maintain_Indexes 947
SQL Server 948
Teradata 949
Oracle 949
DB2 950
Ws_Version_Clear 951
SQL Server 951
Teradata 952
Oracle 953
DB2 954
WsParameterRead 955
SQL Server 955
Teradata 955
Oracle 956
DB2 956
WsParameterReadF 957
SQL Server 957
WsParameterReadG 958
SQL Server 958
Teradata 959
Oracle 959

© Copyright, WhereScape Software Ltd All rights reserved.

DB2 ... 959
WsParameterWrite ... 960
SQL Server ... 960
Teradata ... 961
Oracle ... 961
DB2 ... 961
WsWrkAudit ... 962
SQL Server ... 962
Teradata ... 963
Oracle ... 964
DB2 ... 965
WsWrkAuditBulk ... 966
SQL Server ... 967
Teradata ... 967
Oracle ... 968
DB2 ... 969
WsWrkError ... 970
SQL Server ... 971
Teradata ... 971
Oracle ... 972
DB2 ... 973
WsWrkErrorBulk ... 974
SQL Server ... 975
Teradata ... 975
Oracle ... 976
DB2 ... 977
WsWrkTask ... 978
SQL Server ... 979
Teradata ... 979
Oracle ... 980
DB2 ... 981
Ws_admin_v Views ... 982
Ws_admin_v_audit ... 982
Ws_admin_v_dim_col ... 983
Ws_admin_v_error ... 985
Ws_admin_v_fact_col ... 986
Ws_admin_v_fact_join ... 988
Ws_admin_v_sched ... 988
Ws_admin_v_task ... 992
Retrofitting ... 995
Migrating the Data Warehouse Database Platform ... 995
Importing a Data Model ... 1002
Re-Targeting Source Tables ... 1006
Retro Column Properties ... 1007
General ... 1007
Physical Definition ... 1008

© Copyright, WhereScape Software Ltd All rights reserved.

Meta Definition 1008
Source Details 1009
Retro Column Properties Screen 1010
Retro Column Transformations 1011
Integrating WhereScape RED into an Existing Warehouse 1012
Rebuild 1012
Integrate 1012
Rebuilding 1012
The rebuild process 1012
Integrating 1013
The integrate process 1013
Integrating, Host Scripts 1014
Integrating, Selecting a Table Type 1015
Integrating, Questions 1016
Artificial Key definition 1017
Get Key Function 1017
Update Procedure 1017
Artificial Key Sequence 1018
Update Procedure Definition 1018
Business Key definition 1018
Index definition 1018
Procedure creation 1019
Integrating, Procedures 1019
Parameters 1019
Integrating, Views 1019
Integrating, WhereScape Tables 1020
Relationship Maintenance 1021
Add Relationship 1021
List Relationships 1022
Modify Relationship 1023
Delete Relationship 1023
Generate Relationships 1023
Upgrading RED 1024
WhereScape Versions 1024
Metadata Changes 1024
Application changes 1024
Upgrading RED 1024
Oracle invalid procedures: 1025
Login Checks 1026
Data Source does not match the data warehouse connection's ODBC DSN 1027
More than one connection has the Data Warehouse Indicator Enabled 1029
Oracle Individual User 1030
Data Type Mappings 1031
Using Data Type Mapping Sets 1031
Data Warehouse Connection Properties 1031
Non-Data Warehouse Connection Properties 1032

© Copyright, WhereScape Software Ltd All rights reserved.

Maintaining Data Type Mapping Sets ... 1033
Creating a New Data Type Mapping Set ... 1034
Copying a Data Type Mapping Set ... 1037
Editing a Data Type Mapping Set ... 1039
Deleting a Data Type Mapping Set ... 1041
Loading Custom Data Type Mapping Sets ... 1043
Exporting Custom Data Type Mapping Sets ... 1044
Custom Data Type Mapping Set Examples ... 1046
%length ... 1046
%scale ... 1046
%precision ... 1048
%table_name and %column_name ... 1048
%format ... 1049
%prompt ... 1049
Column Context Menu ... 1052
Properties ... 1052
Change Column(s) ... 1054
Add Column ... 1056
Duplicate Column ... 1057
Delete Column ... 1058
Re-space Order Number ... 1058
Sync Column order with database ... 1059
Impact ... 1060
Send Columns to Another Object ... 1061
Database Functions ... 1063
Using Database Function Sets ... 1063
Column Transformation Properties ... 1063
Transformation Definition ... 1064
Connection Properties ... 1065
Maintaining Database Function Sets ... 1066
Creating a New Database Function Set ... 1068
Copying a Database Function Set ... 1069
Editing a Database Function Set ... 1071
To add a new function to the database function set: ... 1072
To copy an existing function in the database function set: ... 1074
To edit an existing function in the database function set: ... 1075
To delete an existing function in the database function set: ... 1076
To move a function in the database function set up or down in the list: ... 1077
To move a group of functions in the database function set up or down in the list: ... 1078
Deleting a Database Function Set ... 1079
Loading Database Function Sets ... 1081
Exporting Database Function Sets ... 1083
Extended Properties ... 1085
Extended Property Definitions ... 1085
Extended Property Value Assignment ... 1085
Extended Property Lookup ... 1085

© Copyright, WhereScape Software Ltd All rights reserved.

Defining Extended Properties 1086
Creating an Extended Property Definition 1087
Variable Settings 1088
Scope Settings 1088
Maintaining Extended Property Definitions 1089
Extended Property 1089
Function Buttons 1089
Extended Properties Value Assignment 1090
Setting Up Extended Property Values for a Connection 1090
Connection Extended Properties 1090
Setting Up Extended Property Values for an Object 1091
WhereScape RED Object Extended Properties 1091
Extended Properties Lookup 1091
Extended Properties Data Migration Between Repositories 1092
Exporting Extended Properties 1092
Loading Extended Properties 1094
Multi Source Processing 1096
Multi Source Functions and Features 1096
Source Mapping Object 1096
Source Mapping Tool 1097
Global Naming of Source Mappings 1098
Independent Execution of Update Procedures 1099
Table Column Properties 1100
Adding Source Mapping to Objects 1101
Drag and Drop 1103
Maintaining Source Column Mappings 1106
Generating Update Procedures for Source Mapping Objects 1109
Bulk Update Procedure Regeneration 1112
Executing Update Procedures via Scheduler 1113
Reverting to Non Source Mapping Object 1115
Listing Source Mapping Objects 1116
List Source Mapping Objects in a Project or Project Group: 1116
List Source Mapping Objects in an Object Group: 1117
Table and Column Comments 1119
Defining Table Comments 1120
Table Properties 1120
Exporting Table Comments 1121
Loading Table Comments 1124
Defining Column Comments 1127
Column Properties 1127
Exporting Column Comments 1128
Loading Column Comments 1131
Importing Comments from an External Source 1134
Table Comments 1134
Column Comments 1134
Viewing the Import/Export Logs 1135

© Copyright, WhereScape Software Ltd All rights reserved.

Host Script Languages 1136
Defining Host Script Languages 1137
Applying Host Script Languages 1138
Template Properties 1138
Host Script Properties 1139
Maintaining Host Script Languages 1140
Host Script Languages 1140
Function Buttons 1140
Adding a Host Script Language Type 1141
Host Script Languages Data Migration Between Repositories 1143
Loading Host Languages 1143
Exporting Host Script Languages 1145
Dedicated Command Line Interface (REDCLI) 1147
RED Command Line Interface 1149
RED Client Command Line (med.exe) 1149
RED Client Command Line Arguments 1149
Batch Documentation Creation 1151
Batch Deployment Application Creation 1151
New Metadata Repository Start up Task Creation 1152
RED Setup Administrator Command Line (adm.exe) 1153
RED Setup Administrator Command Line Arguments 1153
Quick Application 1155
Quick Repository 1155
Quick Language 1156
Quick Validate 1156
Quick Tutorial 1156
Windows Scheduler 1157
Deploy Application 1157

© Copyright, WhereScape Software Ltd All rights reserved.

Overview

Traditionally ,data warehouses take too long to build and are too hard to change. WhereScape RED is an Integrated Development Environment, designed to support the quick and simple building and managing of data warehouses.

It has the flexibility to enable you to build a variety of architectures including:

- enterprise data warehouses
- dimensional data warehouses
- data marts
- user facing views, aggregates and summaries

In all cases, the core values of WhereScape RED are twofold: its rapid building capabilities that enable better data warehouses to be built, faster, and its integrated environment that simplifies management.

As a data warehouse specific tool, WhereScape RED embodies a simple, pragmatic approach to building data warehouses. With WhereScape RED you specify what you want to achieve by dragging and dropping objects to create a meta view, and then let WhereScape RED do the heavy lifting of creating the necessary tables, procedures, documentation, etc. Data warehouse wizards prompt for additional information at critical points to provide the maximum value from the generated objects.

Different data warehousing approaches including agile, prototyping and waterfall are supported by WhereScape RED . Agile developers will find specific functionality has been included to support common agile practices. For developers who are new to data warehousing, or are looking for a fast, pragmatic approach, WhereScape 's Pragmatic Data Warehousing Methodology can be used.

The basic concepts behind WhereScape 's Pragmatic Data Warehousing Methodology are:

- minimize the impact on the source systems
- centralize management within the data warehouse
- store transactional data at the lowest practical grain within the data warehouse
- snapshot, combine and rollup transactional tables to provide additional value
- utilize star schemas, views or cubes for end user access
- allow for incremental loads from day one
- use of an iterative approach
- simplifying the reconciliation of Load tables to source system tables - data is cleaned and transformed in subsequent Stage tables instead
- design the data warehouse independently from the end user tool layer

WhereScape RED supports these concepts to facilitate very rapid delivery of data warehouses. WhereScape RED controls the flow of data from the source systems through transforming and modeling layers to analysis areas.

Different styles of data warehousing (e.g. EDW 3NF, dimensional, etc.) are supported and utilize different objects, but all follow the same basic flow.

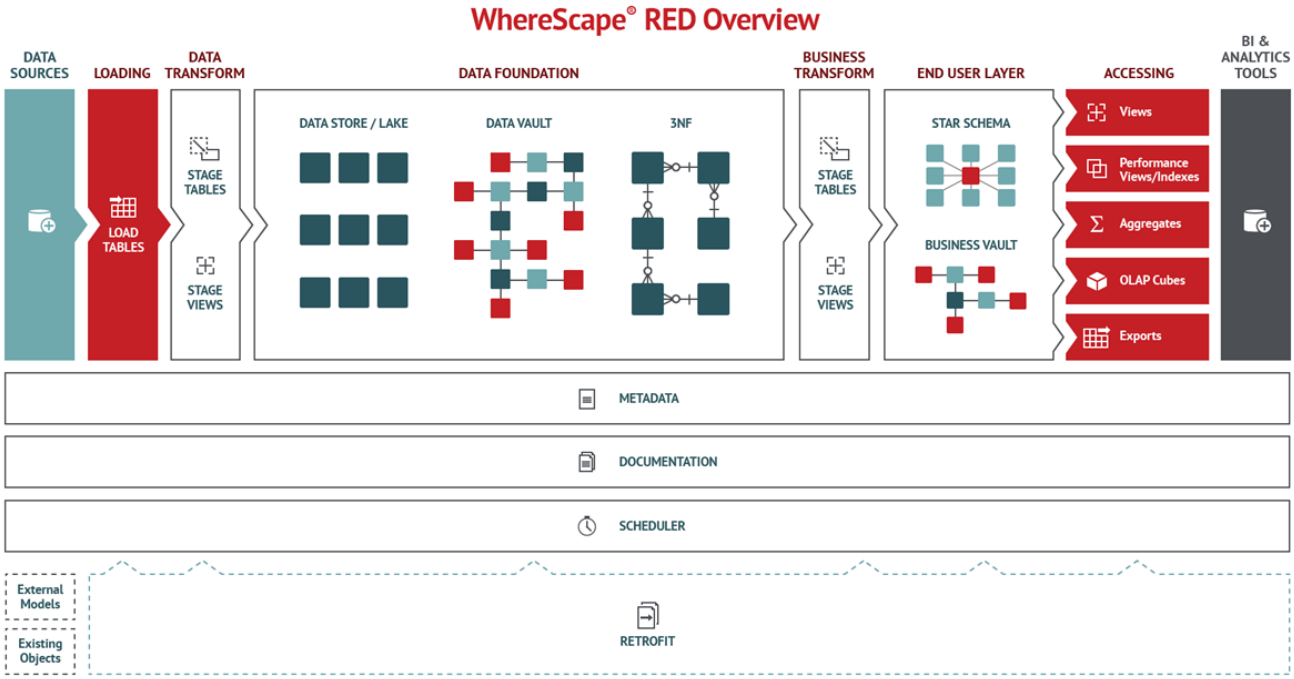
Data Flow - Enterprise Models

1. Source (OLTP) System
2. load tables
3. stage tables
4. data store tables
5. model tables, dimension tables, or detailed (transactional) fact tables
6. roll up fact table(s)
7. aggregate and/or KPI fact table(s)
8. views

© Copyright, WhereScape Software Ltd All rights reserved.

9. export objects
10. Microsoft Analysis Services cubes

The diagram below shows the objects and the information flow:



Data Flow

Data is moved from source tables to load tables via scripts, database links and ODBC links. These load tables are created by dragging and dropping from a connection object. Load tables are generally based on source system tables. Their main purpose is to be a destination for moving data as simply and quickly as possible from the source system. Load tables will generally hold a single unit of data (e.g. last night or last month), and will be truncated at the start of each extract. Transformations can be performed on the columns during the load process if required.

Load tables feed stage tables, which in turn feed data store, model or dimension tables. Data from multiple load tables can be combined at this level.

First tier transactional tables (fact or model) are created and updated from stage tables. Second tier tables (model, summary rollup, aggregate, KPI, etc.) are created and updated from lower level tables.

Cubes can be created from transactional tables or views.

Procedural code

WhereScape RED generates procedural code in the target database's native language (e.g. PL/SQL for Oracle) at each stage in the data warehouse build process. The generated code is, in nearly all cases, sufficient to create a rapid prototype of the data warehouse.

While the generation of code is often seen as a key benefit of WhereScape RED, the ability to control and manage custom code is also critical to the long term management of the data warehouse environment.

In most cases 85-100% of the generated code will be taken through to production with no customization required.

© Copyright, WhereScape Software Ltd All rights reserved.

Scheduler

The flow of data from the source systems to data warehouse tables is controlled and managed by the WhereScape RED scheduler. All generated code includes audit and error logging logic that is used by the scheduler.

The scheduler provides a single point of control for the warehouse. From the scheduler, the state of all jobs can be ascertained. Any warning or error messages can be investigated and if a problem occurs, the scheduler controls the restart of the job from the point of failure.

Documentation

Documenting the data warehouse is often a task left until last, and in many cases done once (if at all!) and not kept up to date. WhereScape RED generates user and technical documentation, including diagrams, in HTML format.

Technical documentation includes copies of all current procedures.

User documentation includes a glossary of business terms, available independently of any end user tool.

Where additional specific information needs to be included in the documentation, WhereScape RED supports the inclusion of custom HTML pages in the generated output. This means in many cases the entire documentation requirements can be managed from one location, and regenerated as changes occur.

WhereScape RED and Traditional ETL Tools

WhereScape RED 's core strength is in the rapid building of data warehouse structures. Organizations that have already purchased traditional ETL tools can use WhereScape RED as a pureplay data warehouse toolset. WhereScape RED can be used to iteratively build data marts or presentation layer objects that need to be constantly updated to keep relevant for end users. In most cases, customers will find that WhereScape RED has enough ETL capabilities to build the entire data warehouse, using the database rather than a proprietary engine to perform ETL processing.

The cross over in functionality between ETL tools and WhereScape RED is not large. WhereScape RED is tightly integrated into the data warehouse database and has an embedded data warehouse building approach. For WhereScape , data movement is the start of the process—from source system to load tables. The key benefits of the product: development productivity and an integrated environment to manage and maintain your data warehouse, comes after the data movement stage. Where a traditional ETL tool is already in use, the output of the ETL process is a WhereScape RED Load, Stage, Dimension, Fact or Model table from which WhereScape RED builds more advanced data warehouse structures.

© Copyright, WhereScape Software Ltd All rights reserved.

Design Introduction

WhereScape RED can be used to build data warehouses based on any number of design philosophies from EDW 3NF enterprise data warehouses with consumer data marts through to federated or conformed star schema based warehouses. In the absence of another approach, the following methodology can be used for the design of data warehouses.

Note:

This section can be skipped if you already have data warehouse design experience or a methodology you wish to utilize. It is meant to provide the novice designer with some tips for designing a data warehouse.

Design Approach

The concepts behind the WhereScape Pragmatic Data Warehouse Methodology are as follows:

1. Building an enterprise-wide data warehouse is a process—an evolution rather than a big bang. Start small and grow the warehouse in manageable chunks until all the pieces are in place. Once you reach that stage, changes and new source systems will continue the process.
2. You need to understand the big picture, but not get lost in it. Talk to all the various departments, business units and companies within the organization. Do so at a relatively high level and try to understand how the information from each area impacts or affects the others. Identify commonalities and areas where the same information is handled in different ways. This process should take days or weeks not months.
3. Identify the high value, high return and possibly easiest areas of the business. Drill down in these areas and break down the workload into small manageable chunks of work, for example, one to two analysis areas. Agree on the first component of the data warehouse and do that.
4. Get an understanding of the source system for this first component or analysis area. If possible, get an entity relationship diagram and talk to the people who built or support the application. Identify the tables that contain the key information you will need. The goal is a quick and initial view, a detailed specification is not required.
5. Design the first component. This design should be a first draft and can be written rather than using a design tool. Remember at this stage what the end users want is not really known, so don't set the design in concrete, or spend a large amount of time in this area.

   Note

   Experienced users of WhereScape RED often dispense with a design and go straight to building a prototype.

6. Build a prototype. In most cases, this should not take more than one or two weeks—experienced WhereScape RED developers can expect to build prototypes in hours or days. Concentrate on the detailed and descriptive data, unless you have a clear picture of the summarized requirements. Do as much as possible in terms of validating the data back to the source system. If dealing with a large or complex source system, then only deliver a segment in this prototype, e.g. one branch, one store, one product group, etc. Keep It Simple.
7. Demonstrate the prototype to a group of the key users. Then drill down to a subset of key users (we recommend no more than three) who will help you go forward with the design. If possible, give these users access to the prototype and get them using the data. Stress that data accuracy is not the issue at this stage, rather the look and feel.
8. Enhance the prototype with the feedback provided by the users. Again a quick process. If complicated requirements evolve, then create a plan to implement, doing the highest value parts first. The goal is to get quick buy in and support from the two or three key users.
9. Provide key users access to the reworked prototype and get them using the data. Have them define the business names for all the measures and attributes and to define any pre-calculated measures that they frequently use. Get them to define the hierarchies in the data. Ascertain the commonly utilized queries and reports, and see if there would be a better way of presenting these.
10. From the user feedback look at the need or possibility of using higher level fact tables, such as summaries, aggregates, snapshot or composite rollup tables.

© Copyright, WhereScape Software Ltd All rights reserved.

The concepts and methodologies for designing and building a data warehouse are beyond the scope of this manual.

It is assumed that the reader understands the basic concepts of a data warehouse, and is familiar with modeling, EDW 3NF, star and snowflake schemas, dimensions, fact tables, etc.

Refer to the WhereScape web site for a basic overview of data warehouse design if required.

© Copyright, WhereScape Software Ltd All rights reserved.

Objects and Windows

WhereScape RED makes use of an object concept when dealing with the different components that make up a data warehouse solution.

The main object types are: Connections, Load Tables, Dimensions, Stage Tables, Fact Tables, OLAP Cubes, Aggregates, Procedures, Host Scripts, Indexes, Retros and Exports.

This chapter explains and provides an overview of each of these object types and how they can be managed and organized. The full functionality of each object is covered in the succeeding chapters.

The various Windows, Panes and Views that form the WhereScape RED tool are also explained.

Object Types

WhereScape RED has a concept of objects which are combined to create real world data warehouses and data marts, fast.

Each WhereScape RED object has properties that enable the data warehouse developer to change how the object is used.

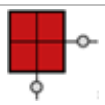
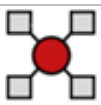
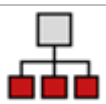
Note:

Some object types may not be available for certain types of WhereScape RED licenses.

WhereScape RED objects include:

Object Type	Icon	Description
Connection		**Connections** define the path to external objects, such as source data. Examples of connection object types are databases, analysis services cubes, operating systems or ODBC sources. Connections isolate environments simplifying, for example, the promotion of code between development and production.
Load Table		**Load tables** are the first entry point of data into the data repository, and typically hold the latest set of change data. These objects contain their definition. Load tables can be defined as database link, ODBC, external, file, script or XML. Based on their definition, they will for example, run a predefined script or create a load script at run time. Pre load actions (e.g. truncate) or post load procedures can be defined as part of a load object. In addition, transformations (either during or after the load) can be defined against columns in a load table.
Dimension Table		**Dimension tables** are the constraining elements in the star schema design and are defined by this object type. WhereScape RED automatically generates procedural code for the three standard types of slowly changing dimensions, as well as date ranged dimensions (where the current version is defined by an external system). WhereScape RED also ships with a standard time dimension which can of course be extended. Dimensions can also be defined as mapping or work tables which do not appear in the generated user documentation.
Dimension View		A **Dimension view** is a database view of a dimension table. It may be a full or partial view. A common usage is to create dimension views where multiple date dimensions exist for one fact table. Other types of views supported by WhereScape RED include fact views, other table views, work views and user defined views.
Stage Tables		**Stage tables** are used in the transformation of raw data into model or star schema format. They typically hold only the latest set of change data. In addition to custom procedures, WhereScape RED can generate different types of procedural code based on the complexity and size of the data set and performance requirements. Examples of procedural types that can be generated are cursor, sorted cursor, set,

© Copyright, WhereScape Software Ltd All rights reserved.

Object Type	Icon	Description
		set + cursor or set merge procedural code. A Stage table can also be defined as a work table, which has the same properties as a stage table but does not appear in the generated user documentation.
EDW 3NF Table		An **EDW 3NF** table is a data warehouse object used to build third normal form enterprise data warehouses. In WhereScape RED, EDW 3NF objects have many of the code generating attributes of stage, dimension and fact tables. Third normal form enterprise data warehouses can be thought of as a source system for star schema data marts. Alternatively, they may be reported off directly by users and reporting tools.
Data Store Table		A **Data Store Table** is a data warehouse object used to store any type of data for later processing. In WhereScape RED, Data Store objects have many of the code generating attributes of stage, dimension and fact tables. Data objects can be thought of as a source system for the data warehouse. Alternatively, they may be reported off directly by users and reporting tools. Data Store Objects can be considered either reference or transactional in nature.
Fact Table		**Fact tables** are the central table in a star schema design. This object type enables the definition of fact tables. They support transactional, rollup, snapshot or partitioned (detail, rollup or exchange) fact tables. Changing a fact table's properties to partitioned, starts a partitioning wizard that prompts for the required information.
KPI Fact Table		This object type supports a special type of fact table. A mandatory KPI (**Key Performance Indicator**) dimension provides a set of KPIs which are stored and maintained by this object type.
Aggregate		The **Aggregate** object type provides a means to speed up access by summarizing data to a higher grain. For dimensional models a rollup of the fact data will allow removal of dimensions that are no longer valid.
View		**View** objects are usually created as end user objects from any table in the data warehouse. The data or columns may be restricted or extra descriptions may be added for use by the end user or reporting tools.
OLAP Cube		The **OLAP Cube** object type uses Analysis Services cubes to deliver OLAP functionality in WhereScape RED . A cube is a set of related measures and dimensions that is used to analyze data from a variety of different front-end tools. OLAP Cubes are built from fact objects and aggregate objects in WhereScape RED.
OLAP Dimension		An **OLAP Dimension** is built by WhereScape RED for every dimension table associated with the fact (or aggregate) table the OLAP Cube is derived from. OLAP Dimensions are shared across one or more OLAP Cubes. In analysis services, a dimension is a group of attributes that represent an area of interest related to the measures in the cube and which are used to analyze the measures in the cube.
Procedure		The **Procedure** object type is used to define and hold database stored procedures. As such it may contain functions, procedures and packages that are generated, modified or custom developed.
Host Script		**Host script** objects are either Windows or UNIX scripts. These scripts are maintained within the WhereScape RED environment and can be scheduled to run in their host environments.
Index		This object type defines database **indexes** used to improve the access times on any of the table object types. (i.e. Load, Stage, Dimension, Fact, Kpi Fact and Aggregate).
Export		**Export** objects are used to manage exports from the data repository. In essence, exports are the reverse of Load tables, taking data from a table to a flat file.

© Copyright, WhereScape Software Ltd All rights reserved.

Object Type	Icon	Description
Retro		**Retros** are used to load predefined data models from modeling tools and to retrofit existing tables into the WhereScape RED metadata.
Retro Copy		**Retros** can be used to copy data from an existing data warehouse into WhereScape RED metadata. Retros can be set as Retro Copy objects to enable data transfer from the existing data warehouse to the new data warehouse.
Template		**Template** objects are used to generate DDL, update/custom procedures and host scripts. Once a template has been created it can be associated with a table and an operation on that table. The template is then used to generate the script used for the associated operation. Each template is assigned a type and a target database, these properties are used to assist with filtering when associating table operations to templates. **Note:** Not all operations support template script generation on all target databases. Utility type templates can contain common code for use by other templates.
Hub		A **Hub** is a table of unique business keys, they usually contain a hash key, business key(s), load date and record source. Hubs should normally have at least one Satellite.
Link		**Links** are many-to-many tables representing current and past relationships between two or more Hub entities and are used to describe associations, transactions, hierarchies and redefinitions of Hub entities in a Data Vault. Links have their own hash key and the hash keys for the Hubs that are linked, as well as a Load Date and Record Source. The attributes describing the context of a link are stored in Satellite Tables.
Satellite		**Satellites** are Data Vault objects that contain metadata which provides context for Hub and Link entities at a given time or over a period of time. Each Satellite entity can contain information on one Hub or Link. Satellite tables contain a hash key for the parent Hub or Link, a timestamp for the date of change and relevant descriptive fields. Satellites are usually created once per source system. Because descriptive attributes can change at different rates, Satellites can also be created based on rate of change.
Source Mapping		**Source mapping** objects can be defined for a number of table object types, for example Stage tables, Hubs, Links and Satellites. They enable these target tables to be sourced from more than one set of source tables. Source mapping objects don't result in data warehouse tables but contain source table and column information metadata, including any transformations and join criteria. This metadata is used to generate a procedure or script to populate the target table. Source mapping objects support more than one insert/update routine from different sources to be defined for the target table, and either executed collectively or independently. Refer to **Multi Source Processing** for details.
Scheduler Jobs		The **Scheduler Jobs** object group lists all the jobs that have been defined in WhereScape RED. Jobs, such as data loads and updates can be run in background mode and/or at a pre-determined time, using the RED Scheduler. Jobs can be added or removed from a project. Refer to **Scheduler** for details

© Copyright, WhereScape Software Ltd All rights reserved.

Object Type	Icon	Description
Parameters		The **Parameters** object group lists all the parameters that have been defined in WhereScape RED. Parameters are a means of passing information between two or more procedures and between the WhereScape RED environment and procedures. Parameters can be added or removed from a project. Refer to **Parameters** for details
Custom1/ Custom2		**Custom1** and **Custom2** objects are user defined objects. These Object Types can be renamed in the **Tools > Options > Object Types > Object Names** menu.

Connections are normally the first objects created. These connections are then used in the creation of Load tables through the drag and drop functionality. Subsequent objects can also be created through the use of drag and drop.

Note:

Although the object types have names that correspond with their primary usage, they can be used for other purposes. For example, the Fact object type could be used to create persistent Stage tables, if required.

Some objects are not supported by all databases, and some advanced properties are specific to the different databases.

© Copyright, WhereScape Software Ltd All rights reserved.

Working with Objects

Most object types perform some form of action in the data warehouse. For example; Dimension, Stage, Fact and Aggregate table based objects are 'Updated' in the data warehouse via the defined update procedure.

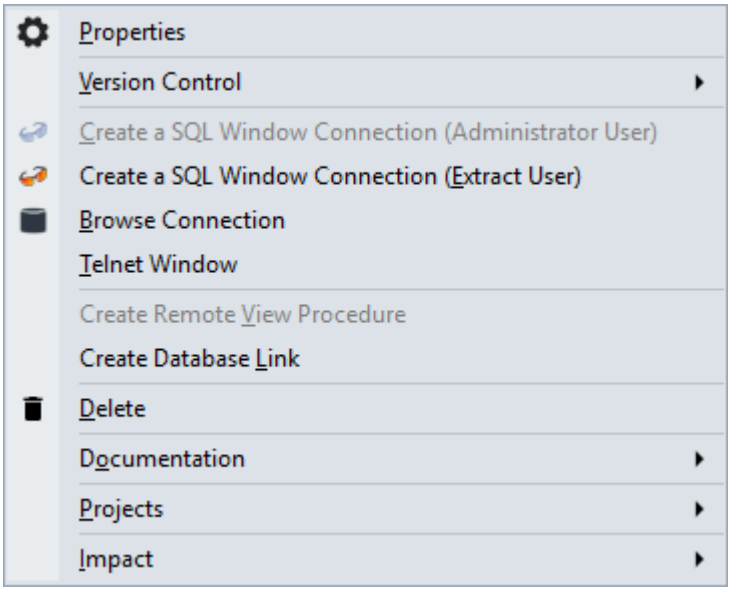
Procedures can be executed in the database. You can right-click objects listed in the left pane of the WhereScape RED **Builder** window to open a context menu which provides a number of options for manipulating the object. This can also be done for objects listed in the middle pane.

Further options are available through the menus provided in the various windows.

The operations of each of the objects is discussed in the sections below.

Connections

Connections once defined, are typically browsed and used as a source for drag and drop operations. For database connections, a database link is normally required. This link can be created via the right-click context menu of the selected connection.



Note:

WhereScape RED supports the use of keyboard shortcuts—the underlined letter of a menu option. For example, pressing **P** opens the **Properties** window of the selected connection, pressing **V** creates a new version of the connection.
Ensure that the **Windows > Control Panel > Ease of Access** setting associated with keyboard shortcuts is enabled, to display keyboard shortcuts in RED. Refer to the relevant MS Windows documentation for details.

Menu Options	Description
Properties	Displays the **Properties** window for the selected connection.
Version Control	A version of a Connection is a copy of the metadata definition of the Connection at the time of the versioning. This version information can be used to create a new Connection object or can simply be left as a backup and reference point. Select this option to perform one of the following: • **New Version** - Enables you to create a new version of the selected Connection. • **Duplicate Object** - Enables you to create a new Connection object as a duplicate of the selected Connection. • **Revert to Version** - Enables you to revert to a previous version of the selected Connection. Displays a list of the available versions of the selected Connection

© Copyright, WhereScape Software Ltd All rights reserved.

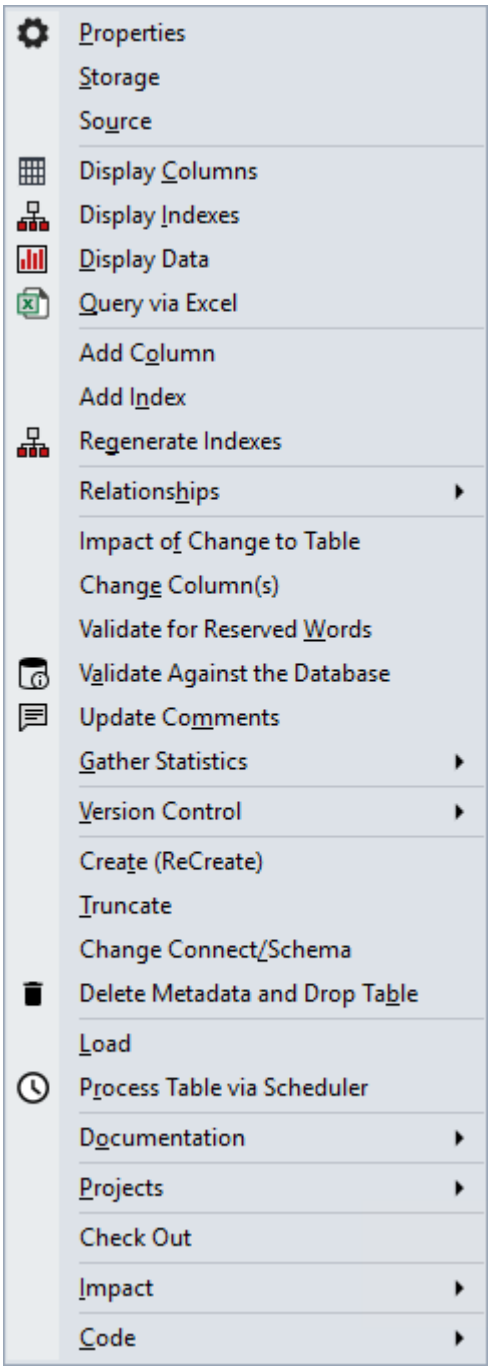
Menu Options	Description
	object, from which you can select and revert.
Create a SQL Window Connection	Opens the RED **SQL Admin** tool which enables you to perform SQL queries in the database connection.
Browse Connection	Displays the **List Source Tables Connection** window which enables you change the properties of the connection you are browsing or switch to another connection. Refer to **Browsing a Connection** for details.
Telnet Window	Creates a Telnet window for UNIX connections.
Create Remote View Procedure	Creates a remote view creation procedure, if required for database connections and loads. Refer to **Remote View Extract** for details.
Create Database Link	Establishes a database link after the relevant connection has been created in RED. Refer to **Creating a Database Link** for details.
Delete	Deletes the meta data of the selected connection, including all related information. A version of the object's meta data is normally auto created (depends on the settings in **Tools > Options > Metadata Versioning**).
Documentation	Select this option to generate (or read if already generated) the WhereScape RED HTML documentation for the selected object. Select this option to perform one of the following: • **Display** the documentation available for the selected object. Refer to **Reading the Documentation** for details. • **Create** the documentation for the selected object. Refer to **Creating Documentation** for details.
Projects	Select this option to perform one of the following: • **Remove from Project** - Select to remove the Connection from a project. Refer to **Removing Objects from Projects** for details. • **Add to Project** - Select to add the Connection to a project. Refer to **Adding Objects to Projects** for details. • **List Projects** - Displays a list of projects which contain the current object, results are shown in the bottom pane. Refer to **List Project Memberships for Objects** for details. Refer to **Organizing Objects** for details. **Notes:** • Multiple objects can be selected by double-clicking the Object Group icon in the left pane and then Ctrl + clicking multiple objects in the Drop Target (middle) pane. • If there aren't any projects in the repository, the above options are unavailable.
Impact	Select this option to run reports on Load and Export objects associated with the connection. Select this option to perform one of the following: • **Load Tables Report** • **Export Tables Report** Refer to **Table Reports** for details.
Build	Select this option to build an application based on the connection(s) selected from the middle pane. Refer to **Building Applications from Object Groups** for details. This option is available from the context menu of connections listed in the middle pane.

© Copyright, WhereScape Software Ltd All rights reserved.

Load Tables

Load tables once defined would normally be created and loaded, unless these actions were performed as part of the drag and drop operation.

The menu below shows the operations that can be performed on Load tables.



Note:

WhereScape RED supports the use of keyboard shortcuts—the underlined letter of a menu option. For example, pressing **P** opens the **Properties** window of the selected table, pressing **L** performs an interactive load, etc.
Ensure that the **Windows > Control Panel > Ease of Access** setting associated with keyboard shortcuts is enabled, to display keyboard shortcuts in RED. Refer to the relevant MS Windows documentation for details.

© Copyright, WhereScape Software Ltd All rights reserved.

Menu Options	Description
Properties	Displays the **Properties** window for the Load table, albeit focused on different tabs within this window.
Storage	Displays the **Properties** window for the Load table, albeit focused on different tabs within this window.
Source	Displays the **Properties** window for the Load table, albeit focused on different tabs within this window.
Display Columns	Displays columns of the load table.
Display Indexes	Displays indexes of the load table.
Display Data	Displays data of the load table. **Tip:** If the data is displayed, only the first 100 rows are returned from the table. Either the SQL Admin tool (accessible via the WhereScape start menu option), or the Excel query must be used if more detailed data analysis is required.
Query via Excel	Click to query columns in Microsoft Excel. **Tip:** When a column list has been displayed in the central pane, it is sorted based on the **order** field associated with each column. Clicking the column label **Col name** will sort the columns into alphabetical order. A subsequent click will re-sort based on the **order** field.
Add Column	Select this option to add new columns. Normally columns can be added via drag and drop.
Add Index	Select this option to add new indexes. Normally indexes are created during the procedure generation phase.
Regenerate Indexes	Select this option to add missing standard indexes. Selecting this option displays a window with options to regenerate missing indexes in the metadata and recreate them or to just regenerate the missing indexes in the metadata.
Relationships	Select this option to manage enhanced relationships. Select this option to perform one of the following: • **Add Relationship** - Displays the Add Relationships dialog. • **List Relationships** - Displays a list of enhanced relationships in the **Drop Target Pane** for the selected object. • **Generate Relationships** - Generates relationships which have not yet been defined in the metadata
Impact of Change to Table	Select this option to produce a list of objects that will be potentially impacted by a change to the load table structure.
Change Column(s)	Select this option to apply changes to a selected number of columns
Validate for Reserve Words	Select this option to produce a list of table or column names where reserved words have been used; enabled for supported ODBC Drivers.
Validate Against the database	Select this option to compare the metadata for the load table with the physical table resident in the database, and where required the table is altered to match the metadata.
Update Comments	Select this option to refresh table and column comments on the table, using the table's description and columns' business definition. **Note:** You can manage the table and column comments outside the data warehouse environment via the **Table and Column Comments** menu option.

© Copyright, WhereScape Software Ltd All rights reserved.

<table>
<tr><th>Menu Options</th><th>Description</th></tr>
<tr><td>Gather Statistics</td><td>
Select this option to gather statistics on a table. This action enables the underlying database to optimize each query based on the statistics collected about the data that is being accessed. Select this option to perform one of the following:

Perform Full Statistics
Perform Quick (Sample) Statistics

Gathering statistics can be performed on any table by selecting this option from a table's right click menu, or to automate this process, by adding a statistics task to a job being processed by the scheduler (Stats, Quick Stats, Analyze or Quick Analyze). For more information about adding statistics tasks to jobs for SQL Server, Oracle and DB2 refer to Editing Tasks in a Job.

Tip:

Oracle has the statistics option in the Options window to save a generic statistics statement for analyze, quick analyze, stats and quick stats. If there is no statistics statement saved then a default statistics statement is used. The statistics process from the object context menu for tables is described below:
<table>
<tr><th colspan="2">Oracle Tables</th></tr>
<tr><td>Perform Full Statistics</td><td>BEGIN dbms_stats.gather_table_stats(ownname=>'schema', tabname=>'table' , cascade=>TRUE); END;</td></tr>
<tr><td>Perform Quick (Sample) Statistics</td><td>BEGIN dbms_stats.gather_table_stats(ownname=>'schema', tabname=>'table' , estimate_percent=>'3' , cascade=>TRUE); END;</td></tr>
<tr><td>Perform Full Analyze</td><td>ANALYZE TABLE schema.table COMPUTE STATISTICS</td></tr>
<tr><td>Perform Quick (Sample) Analyze</td><td>ANALYZE TABLE schema.table ESTIMATE STATISTICS SAMPLE 3 PERCENT</td></tr>
<tr><th colspan="2">SQL Server Tables</th></tr>
<tr><td>Perform Full Statistics</td><td>UPDATE STATISTICS schema.table WITH FULLSCAN</td></tr>
<tr><td>Perform Quick (Sample) Statistics</td><td>UPDATE STATISTICS schema.table WITH SAMPLE 3 PERCENT</td></tr>
<tr><th colspan="2">DB2 Tables</th></tr>
<tr><td>Perform Full Statistics</td><td>RUNSTATS ON TABLE schema.table WITH DISTRIBUTION AND DETAILED INDEXES ALL</td></tr>
<tr><td>Perform Quick (Sample) Statistics</td><td>RUNSTATS ON TABLE schema.table</td></tr>
<tr><td>Perform Full Analyze</td><td>RUNSTATS ON TABLE schema.table WITH DISTRIBUTION AND DETAILED INDEXES ALL</td></tr>
<tr><td>Perform Quick (Sample) Analyze</td><td>RUNSTATS ON TABLE schema.table</td></tr>
</table>
Oracle Gather Statistics context menu example:

Perform Full Statistics
Perform Quick (Sample) Statistics
Perform Full Analyze
Perform Quick (Sample) Analyze

To add specific commands for Oracle statistics performed by the scheduler, refer to Statistics for details.
</td></tr>
</table>

© Copyright, WhereScape Software Ltd All rights reserved.

Menu Options	Description
Version Control	A version of a Load table is a copy of the metadata definition of the table at the time of the versioning. This version information can be used to create a new Load table or can simply be left as a backup and reference point. Select this option to perform one of the following: • **New Version** - Enables you to create a new version of the selected Load table. • **Build Application** - Enables you to build an application file for the selected Load table. • **Duplicate Object** - Enables you to create a new Load table as a duplicate of the selected table. • **Revert to Version** - Enables you to revert to a previous version of the selected Load table. Displays a list of the available versions of the selected Load table, from which you can select and revert. **Tip:** The extended property values set in the original Load table is copied to the duplicate Load table. Refer to **Extended Properties** for details.
Create (ReCreate)	Select this option to create the table in the database based on the definition stored in the metadata. To alter a table select the Validate against database option. Refer to section on table validation for details.
Truncate	Select this option to truncate the table.
Change Connect/ Schema	This option enables the rapid changing of the Connection information associated with the Load table. This information can be changed in-bulk for a number of Load tables. Refer to **Changing Load Connection and Schema** for details.
Delete Metadata and Drop Table	Select this to option perform one of the following delete options: • **Delete metadata and drop object** - This option deletes the metadata definition for the table and drops the table object from the database (default). This is a permanent delete and no recovery is provided, use with caution. • **Delete metadata only** - This option deletes the metadata definition for the table. A version of the object's metadata is normally auto created (depends on the settings in **Tools > Options > Metadata Versioning**).
Load	Select this option to perform an interactive load of the data. The method of loading depends on the type of connection. This menu option is intended for use with small data volumes as in a prototype environment. Large data volumes would normally be scheduled. The Load locks the WhereScape RED screen until completed. **Tips:** • For ODBC based loads in an Oracle data warehouse, this interactive load does not use Direct Path loading so will be slower than a scheduled load. • The load option does not drop or create any indexes. Use the Process option if indexes need to be maintained.
Process Table via Scheduler	Select this option to send a request to the Scheduler to immediately process the Load table. This process drops any indexes marked as pre_drop, load the data and rebuild any required indexes. Control is immediately returned to the user and the loading occurs via the Scheduler.

© Copyright, WhereScape Software Ltd All rights reserved.

Menu Options	Description
Documentation	Select this option to generate (or read if already generated) the WhereScape RED HTML documentation for the selected object. Select this option to perform one of the following: • **Display** the documentation available for the selected object. Refer to **Reading the Documentation** for details. • **Create** the documentation for the selected object. Refer to **Creating Documentation** for details.
Projects	Select this option to perform one of the following: • **Remove from Project** - Select to remove the Load table from a project. Refer to **Removing Objects from Projects** for details. • **Add to Project** - Select to add the Load table to a project. Refer to **Adding Objects to Projects** for details. • **List Projects** - Displays a list of projects which contain the current object, results are shown in the bottom pane. Refer to **List Project Memberships for Objects** for details. **Notes:** • Multiple objects can be selected by double-clicking the Object Group icon in the left pane and then Ctrl + clicking multiple objects in the Drop Target (middle) pane. • If there aren't any projects in the repository, the above options are unavailable.
Check Out	Select this option to check out the object for editing and prevent any other users from being able to modify, update or delete any of their associated objects while you are making changes. Refer to **Object Check-Outs and Check-Ins** for details.
Impact	Select this option to produce a number of reports and diagrams. Select this option to perform one of the following: • **Track Back Report** • **Track Forward Report** • **Track Back Diagram** • **Track Forward Diagram** • **Dependent Jobs Report**
Code	Select this option to view a procedure associated to a table or to regenerate the table's update procedure. Select this option to perform one of the following: • **View Load script** • **View Post Load script** • **Regenerate Load script** • **Rebuild Post Load script** • **Regenerate Post Load script** **Notes:** • Only Load tables with one or more defined procedures have the Code view option. • If the Load table is Script-based load type, either option is displayed in the Code sub menu: o Generate Load Script (if a script is not yet generated) o Regenerate <script name> (if a script has already been generated). Refer to **Script based load** for details.

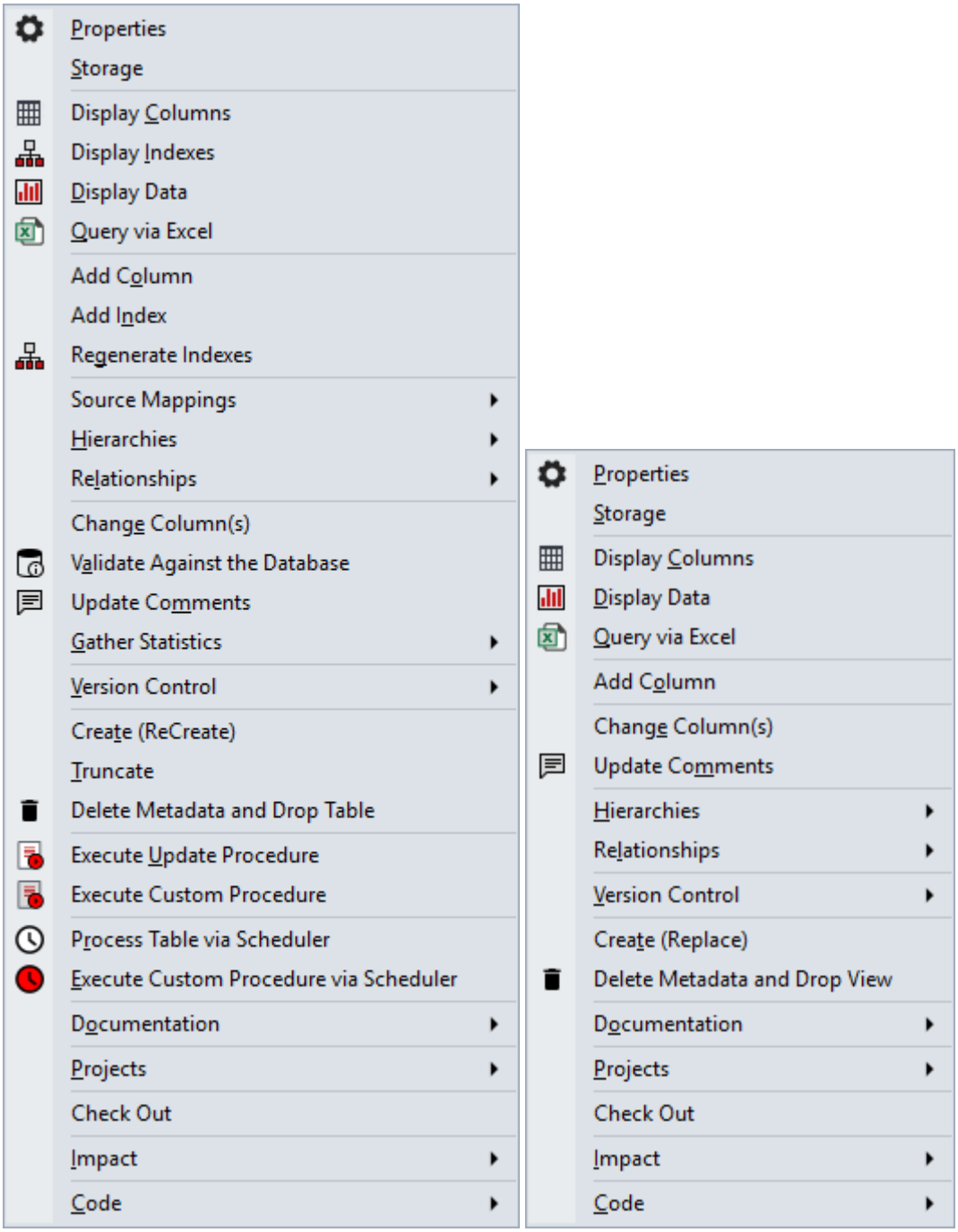
© Copyright, WhereScape Software Ltd All rights reserved.

Menu Options	Description
Build	Select this option to build a Scheduler job or an application based on the Load table(s) selected from the middle pane. Refer to **Building Scheduler Jobs from Object Groups** and **Building Applications from Object Groups** for details. This option is available from the context menu of Load tables listed in the middle pane.

© Copyright, WhereScape Software Ltd All rights reserved.

Dimension Tables

The context menu for **Dimensions** is shown below (dimension **tables** on the left, dimension **views** on the right):



Note:

WhereScape RED supports the use of keyboard shortcuts—the underlined letter of a menu option. For example, pressing **P** opens the **Properties** window of the selected table, pressing **D** displays data, etc. Ensure that the **Windows > Control Panel > Ease of Access** setting associated with keyboard shortcuts is enabled, to display keyboard shortcuts in RED. Refer to the relevant MS Windows documentation for details.

Tip:

The bulk of the menu options are same as for **Load tables**.

© Copyright, WhereScape Software Ltd All rights reserved.

Menu Options	Description
Source Mappings	Source mapping objects are created when there is a requirement to have more than one set of source tables to populate the target table and provides for the generation of an update and custom procedure for each source mapping object. Select this option to perform one of the following: • **Add Source Mapping** - Select to add a source mapping. • **List Source Mappings** - Select to list a source mapping. Refer to **Multi Source Processing** for details.
Hierarchies	Select this option to perform one of the following: • **Add Hierarchy** - Select to add hierarchies. • **List Hierarchies** - Select to list hierarchies. • **List Hierarchy Elements** - Select to list hierarchy elements. • **Copy Hierarchies from Source** - Select to copy all hierarchies from the source table to the destination table. Source hierarchies are copied to the destination table automatically during table creation, but this feature is useful if the source table has been updated since destination table was created. Refer to **Dimension Hierarchies** for details.
Relationships	Select this option to perform one of the following: • **Add Relationship** - Select to add relationships. • **List Relationships** - Select to list relationships. • **Generate Relationships** - Select to generate relationships. Refer to **Relationship Maintenance** for details.
Execute Update Procedure	Select this option to execute the procedure defined as the 'update procedure' for the table. The procedure is executed interactively and locks the screen until completed. This menu option is only intended for use when working with small/prototype data volumes, and no index handling is performed.
Execute Custom Procedure	Select this option to execute the procedure defined as the 'custom procedure' for the table. As with the update procedure, the custom procedure is executed interactively.
Process Table via Scheduler	Select this option to process the selected table via the WhereScapeRED scheduler. Refer to **Scheduler** for details.
Execute Custom Procedure via Scheduler	Select this option to execute the procedure defined as the 'custom procedure' for the table via the WhereScape RED scheduler. Refer to **Scheduler** for details.
Code	Select this option to view a procedure attached to a table or to rebuild the table's update procedure; and to view or rebuild the Get Key function on the Dimension/Dimension View. **Tip:** The Get Key function is not available for dimension objects that are stored in custom database targets. Select this option to perform one of the following: • View update_dim_customer • View custom_dim_customer • View get_dim_customer_key • Rebuild update_dim_customer • Regenerate update_dim_customer • Rebuild custom_dim_customer • Regenerate custom_dim_customer • Rebuild get_dim_customer_key **Note:** Only tables with one or more defined procedures have the Code option.

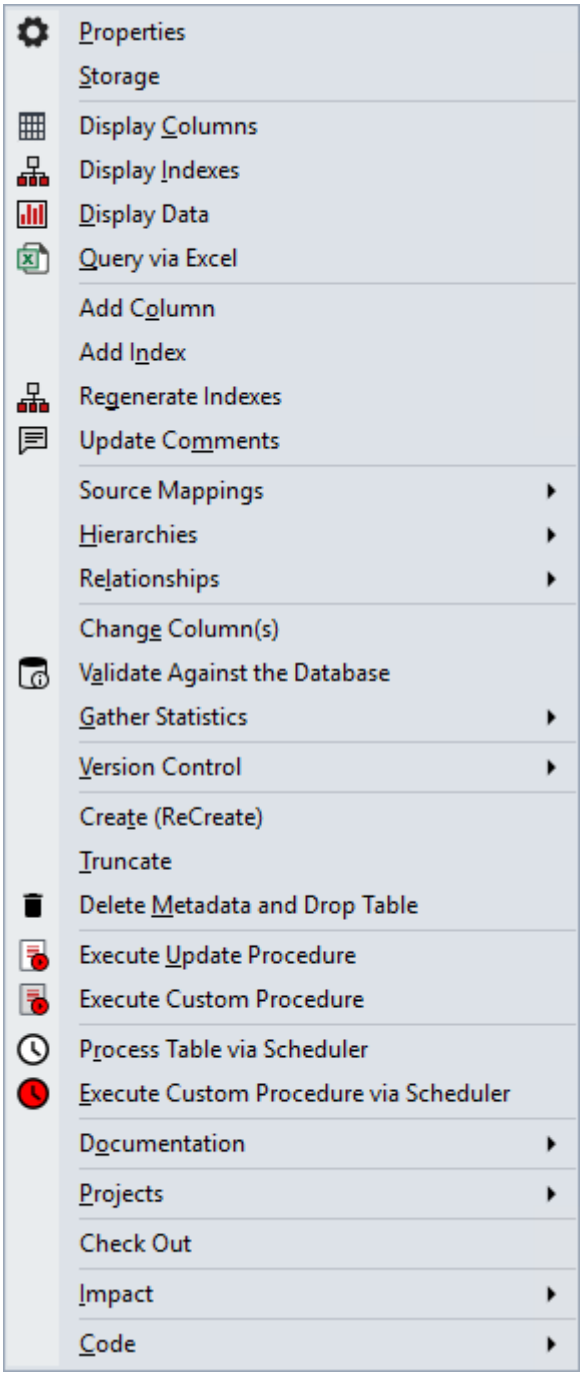
© Copyright, WhereScape Software Ltd All rights reserved.

Tip:	
Build	Select this option to build a Scheduler job or an application based on the Dimension table(s) selected from the middle pane. Refer to **Building Scheduler Jobs from Object Groups** and **Building Applications from Object Groups** for details. This option is available from the context menu of Dimension tables listed in the middle pane.

© Copyright, WhereScape Software Ltd All rights reserved.

Data Store Tables

Following is an example of a standard menu for Data Store tables.



Note:

WhereScape RED supports the use of keyboard shortcuts—the underlined letter of a menu option. For example, pressing **P** opens the **Properties** window of the selected table, pressing **D** displays data, etc. Ensure that the **Windows > Control Panel > Ease of Access** setting associated with keyboard shortcuts is enabled, to display keyboard shortcuts in RED. Refer to the relevant MS Windows documentation for details.

Tip:

The menu options available for Data Store tables is a subset of the options for **Dimension Tables** including the Code menu options.

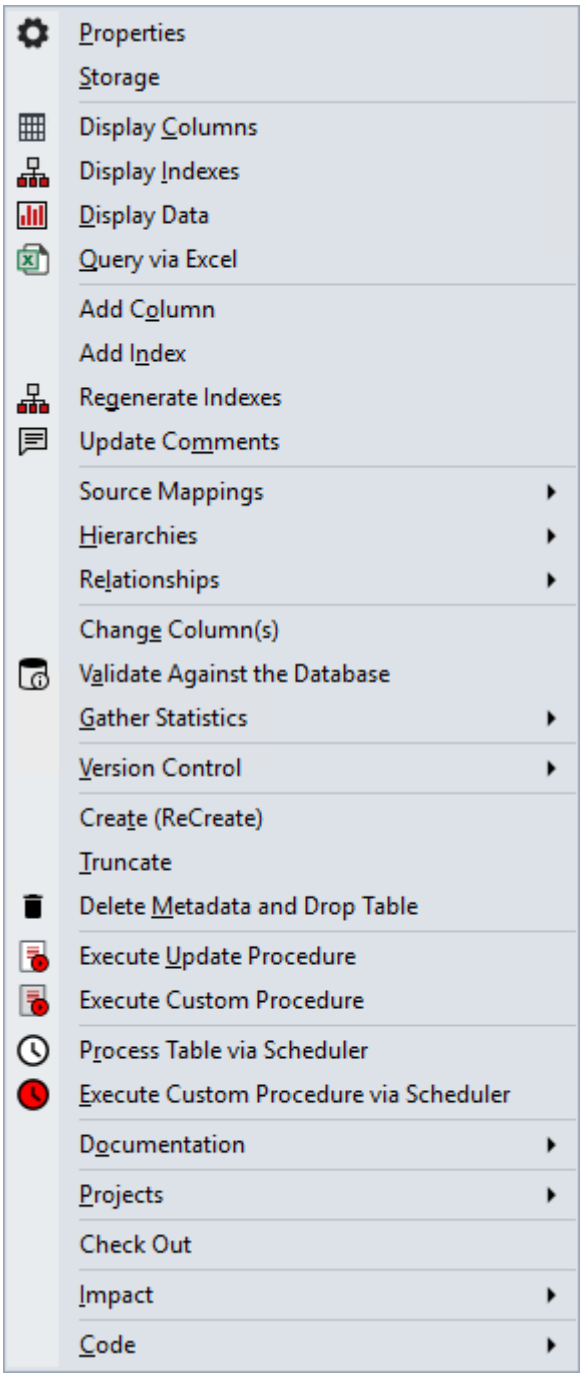
© Copyright, WhereScape Software Ltd All rights reserved.

Menu Options	Description
Code	Select this option to view a procedure attached to a table or to rebuild or regenerate the table's update procedure. Select this option to perform one of the following: • View update_ds_product • View custom_ds_product • Rebuild update_ds_product • Regenerate update_ds_product • Rebuild custom_ds_product • Regenerate custom_ds_product • Display Procedure Build Options Choose a procedure from the list to open in the procedure editor in view mode or choose to rebuild or regenerate the update procedure.
Build	Select this option to build a Scheduler job or an application based on the Data Store table(s) selected from the middle pane. Refer to **Building Scheduler Jobs from Object Groups** and **Building Applications from Object Groups** for details. This option is available from the context menu of Data Store tables listed in the middle pane.

© Copyright, WhereScape Software Ltd All rights reserved.

EDW 3NF Tables

Following is an example of a standard menu for EDW 3NF Tables.



Note:

WhereScape RED supports the use of keyboard shortcuts—the underlined letter of a menu option. For example, pressing **P** opens the **Properties** window of the selected table, pressing **D** displays data, etc. Ensure that the **Windows > Control Panel > Ease of Access** setting associated with keyboard shortcuts is enabled, to display keyboard shortcuts in RED. Refer to the relevant MS Windows documentation for details.

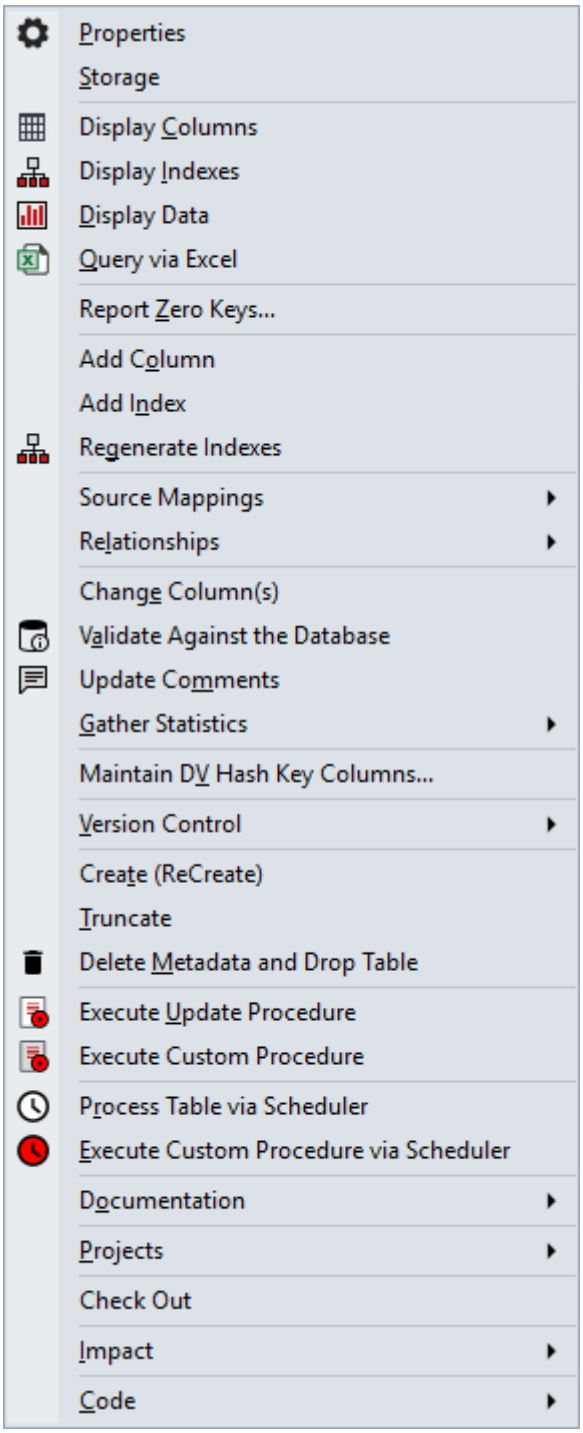
Tip:

The menu options available for Data Store tables is a subset of the options for **Data Store Tables**.

© Copyright, WhereScape Software Ltd All rights reserved.

Stage Tables

Following is an example of a standard menu for Stage tables.



Note:

WhereScape RED supports the use of keyboard shortcuts—the underlined letter of a menu option. For example, pressing **P** opens the **Properties** window of the selected table, pressing **D** displays data, etc. Ensure that the **Windows > Control Panel > Ease of Access** setting associated with keyboard shortcuts is enabled, to display keyboard shortcuts in RED. Refer to the relevant MS Windows documentation for details.

© Copyright, WhereScape Software Ltd All rights reserved.

Tip:

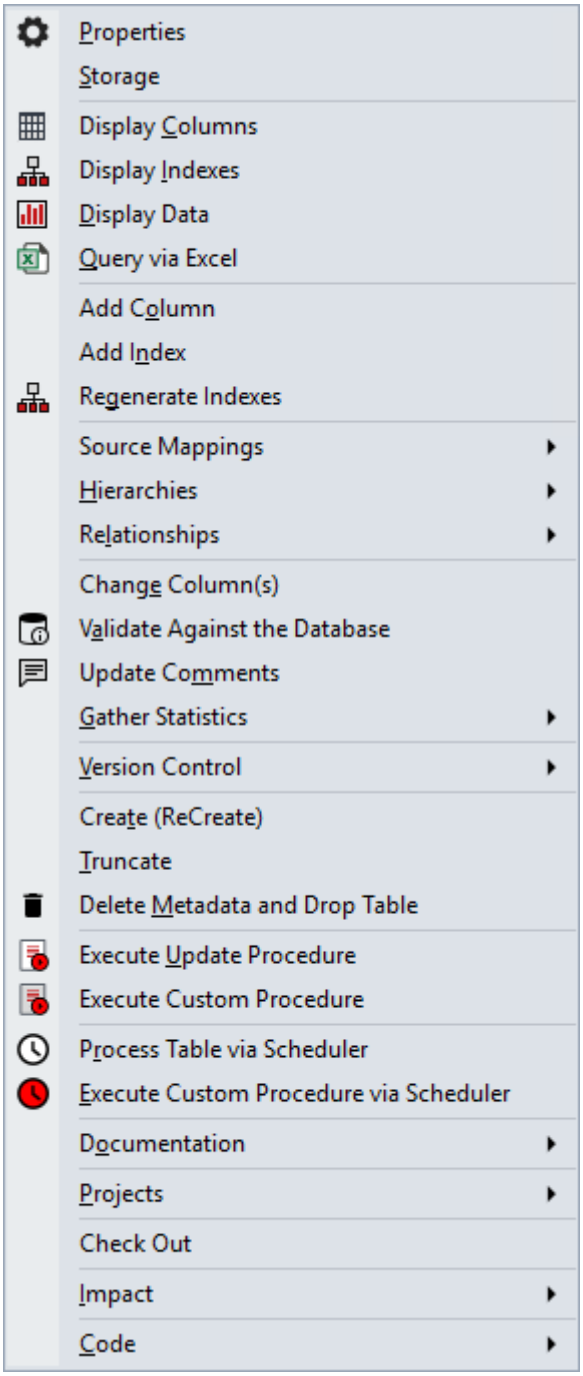
The menu options available for Stage tables is a subset of the options for **Data Store Tables**, except for the **Report Zero Keys...** element.

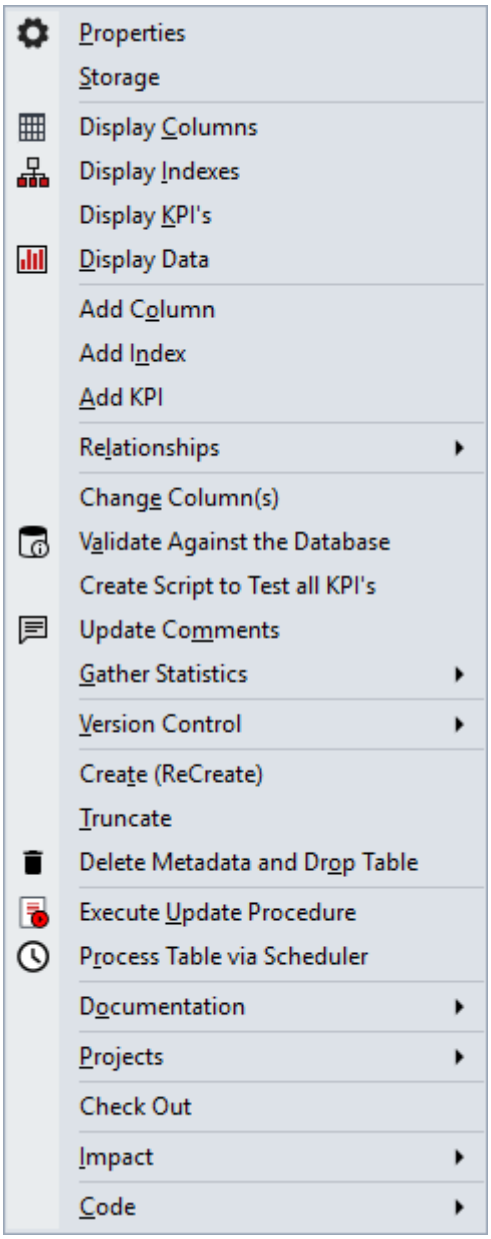
Menu Options	Description
Report Zero Keys...	Select this option to initiate the report Records that Failed a Dimension Join. Refer to **Records that Failed a Dimension Join** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

Fact Tables/KPI Fact Tables

Following is an example of a standard menu for Fact Tables and KPI Fact tables.





Note:

WhereScape RED supports the use of keyboard shortcuts—the underlined letter of a menu option. For example, pressing **P** opens the **Properties** window of the selected table, pressing **D** displays data, etc. Ensure that the **Windows > Control Panel > Ease of Access** setting associated with keyboard shortcuts is enabled, to display keyboard shortcuts in RED. Refer to the relevant MS Windows documentation for details.

Tip:

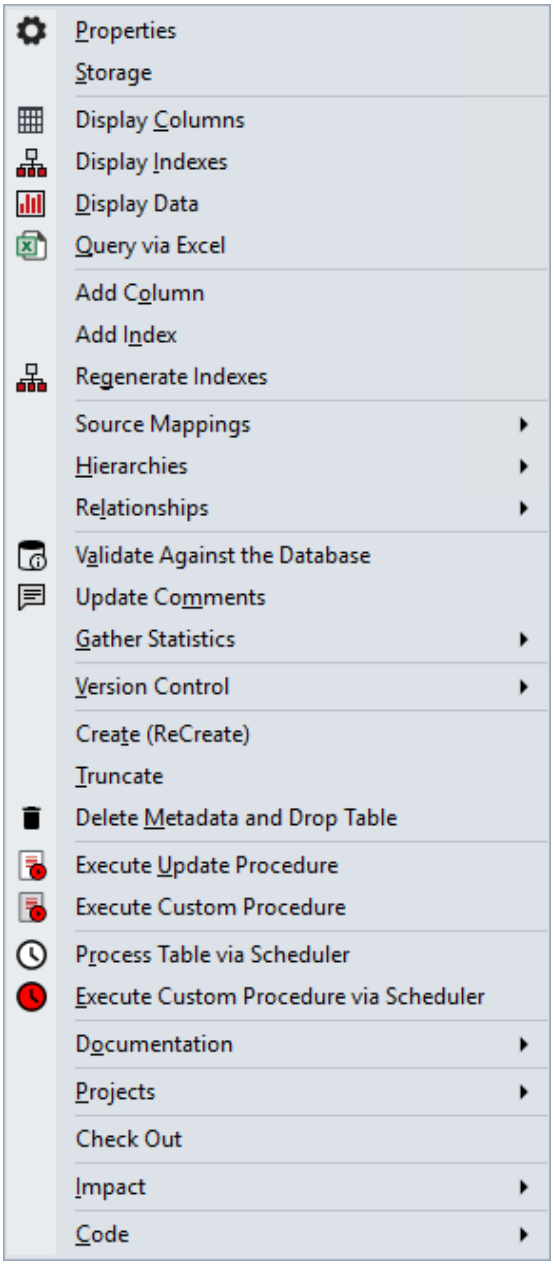
The menu options available for Fact Tables are a subset of the options for **Data Store Tables**. KPI Fact Tables have three additional menu options as mentioned below.

© Copyright, WhereScape Software Ltd All rights reserved.

Menu Options	Description
Display KPI's	Select this option to display all KPIs that have been set up for the KPI fact table in the Drop Target pane (middle pane).
Add KPI's	Select this option to add a new KPI.
Create Script To Test All KPI's	Select this option to generate an SQL script to test the KPI definitions for a specified time window.

Aggregate Tables

Following is an example of a standard menu for Aggregate tables.



Note:

WhereScape RED supports the use of keyboard shortcuts—the underlined letter of a menu option. For example, pressing **P** opens the **Properties** window of the selected table, pressing **D** displays data, etc. Ensure that the **Windows > Control Panel > Ease of Access** setting associated with keyboard shortcuts is enabled, to display keyboard shortcuts in RED. Refer to the relevant MS Windows documentation for details.

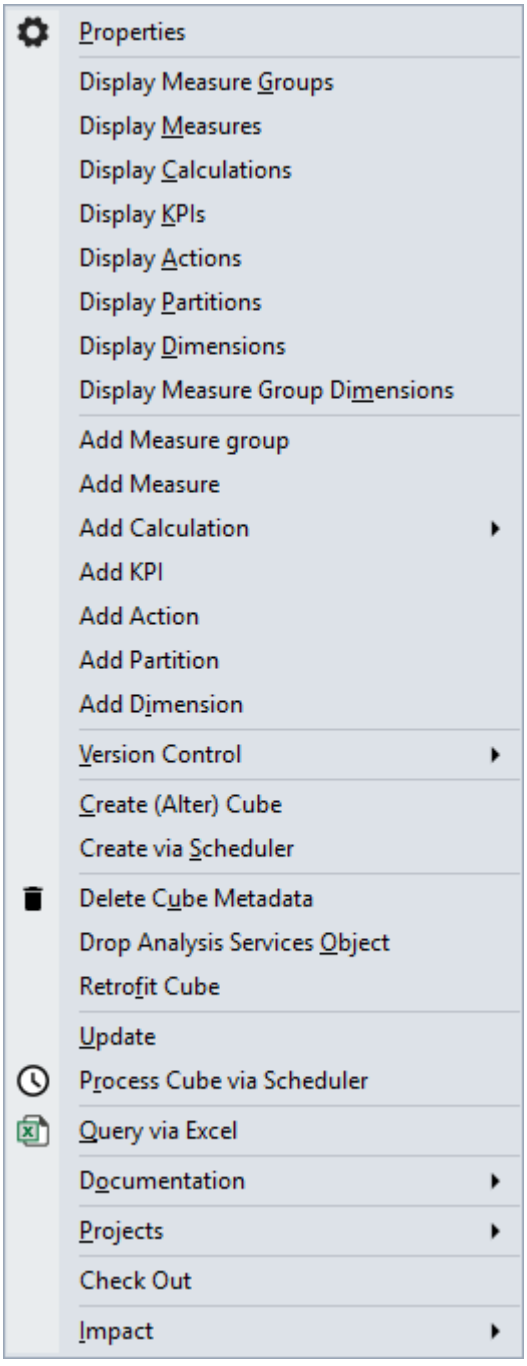
© Copyright, WhereScape Software Ltd All rights reserved.

Tip:

The menu options available for Aggregate tables is a subset of the options for **Data Store Tables**.

OLAP Cubes

Following is an example of a standard menu for OLAP Cubes.



Note:

WhereScape RED supports the use of keyboard shortcuts—the underlined letter of a menu option. For example, pressing **P** opens the **Properties** window of the selected table, pressing **D** displays dimensions, etc.
Ensure that the **Windows > Control Panel > Ease of Access** setting associated with keyboard shortcuts is enabled, to display keyboard shortcuts in RED. Refer to the relevant MS Windows documentation for details.

© Copyright, WhereScape Software Ltd All rights reserved.

Menu Options	Description
Properties	Select this option to display the **Properties** window that defines cube creation options and access to documentation tabs.
Display Measure Groups	Select this option to display the details of the measure groups associated with the cube in the Drop Target (middle) pane.
Display Measures	Select this option to list the measures associated with the measure groups in the cube. This is the default view in the Drop Target (middle) pane when a cube is selected in the left pane with a single click.
Display Calculations	Select this option to list all the calculated members defined in the cube
Display KPIs	Select this option to list all the Key Performance Indicators defined in the cube
Display Actions	Select this option to list all the actions defined in the cube.
Display Partitions	Select this option to list all the partitions defined against the related measure groups within the cube.
Display Dimensions	Select this option to list all the dimensions defined in the cube.
Display Measure Group Dimensions	Select this option to display a cross tab report in the Drop Target (middle) pane showing cube dimensions relating cube measure groups.
Add Measure Group	Select this option to add a new measure group to the cube.
Add Measure	Select this option to add another measure to the cube.
Add Calculation	Select this option to add a new calculated member to the cube.
Add KPI	Select this option to add a new KPI to the cube.
Add Action	Select this option to add a new action to the cube.
Add Partition	Select this option to add a new partition to a measure group in the cube.
Add Dimension	Select this option to add an existing OLAP Dimension to the cube.
Version Control	A version of an OLAP cube is a copy of the metadata definition of the cube at the time of the versioning. This version information can be used to create a new OLAP cube or can simply be left as a backup and reference point. Select this option to perform one of the following: • **New Version** - Enables you to create a new version of the OLAP cube. • **Build Application** - Enables you to build an application file for the OLAP cube.
Create (Alter) Cube	Select this option to create the cube and supporting objects in Analysis Services (including cube database, data source view (DSV) and dimensions) based on the definition in WhereScape RED.
Create via Scheduler	Select this option to submit a create of the OLAP Cube via the Scheduler.
Delete Cube Metadata	Deletes the meta data of the selected OLAP cube, including all related information. A version of the object's metadata is normally auto created (depends on the settings in **Tools > Options > Metadata Versioning**).
Drop Analysis Services object	Select this option to drop an object associated with the OLAP cube in Analysis Services.
Retrofit Cube	Select this option to retrofit the OLAP cube from Analysis Services.
Update	Select this option to process the cube in Analysis Services interactively from the WhereScape RED interface.
Process Cube via Scheduler	Select this option to generate a WhereScape RED scheduler job to process the cube in Analysis Services.
Query via Excel	Select this option to open up an .oqy file in Microsoft Excel. **Note:** Due to a limitation in the Microsoft Office installation it may be necessary to associate the **.oqy** file extension with Microsoft Excel before this option will succeed.

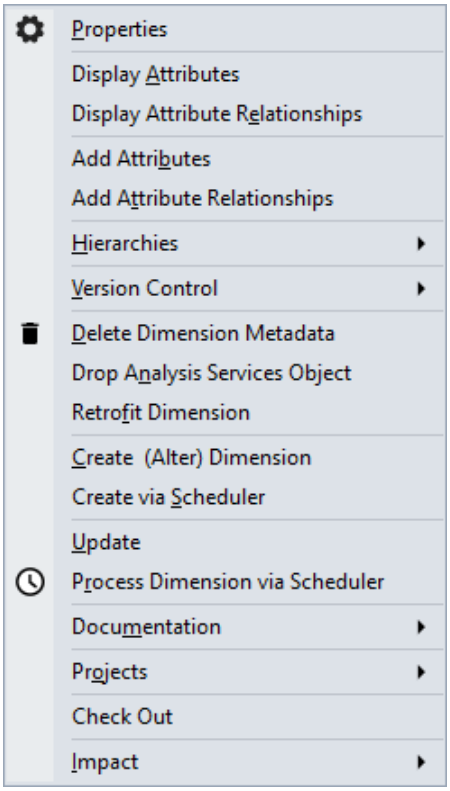
© Copyright, WhereScape Software Ltd All rights reserved.

Note:	
Documentation	Select this option to generate (or read if already generated) the WhereScape RED HTML documentation for the selected object. Select this option to perform one of the following: • **Display** the documentation available for the selected object. Refer to **Reading the Documentation** for details. • **Create** the documentation for the selected object. Refer to **Creating Documentation** for details.
Projects	Select this option to perform one of the following: • **Remove from Project** - Select to remove the cube from a project. Refer to **Removing Objects from Projects** for details. • **Add to Project** - Select to add the cube to a project. Refer to **Adding Objects to Projects** for details. • **List Projects** - Displays a list of projects which contain the current object, results are shown in the bottom pane. Refer to **List Project Memberships for Objects** for details. **Note:** Multiple objects can be selected by double-clicking the Object Group icon in the left pane and then Ctrl + clicking multiple objects in the Drop Target (middle) pane.
Check Out	Select this option to check out the object for editing aand prevent any other users from being able to modify, update or delete any of their associated objects while you are making changes. Refer to **Object Check-Outs and Check-Ins** for details.
Impact	Select this option to produce a Track Back diagram on the OLAP cube. Select this option to perform one of the following: • **Track Back Report** • **Track Forward Report** • **Track Back Diagram** • **Track Forward Diagram** • **Dependent Jobs Report**

© Copyright, WhereScape Software Ltd All rights reserved.

OLAP Dimensions

Following is an example of a standard menu for an OLAP Dimension.



Note:

WhereScape RED supports the use of keyboard shortcuts—the underlined letter of a menu option. For example, pressing **P** opens the **Properties** window of the selected table, pressing **D** deletes the dimension metadata, etc.
Ensure that the **Windows > Control Panel > Ease of Access** setting associated with keyboard shortcuts is enabled, to display keyboard shortcuts in RED. Refer to the relevant MS Windows documentation for details.

Menu Options	Description
Properties	Select this option to display the OLAP Dimension Properties dialog which includes documentation tabs.
Display Attributes	Select this option to list the attributes for the selected dimension. This is the default view when an OLAP Dimension is selected in the left pane.
Display Attribute relationships	Select this option to display the relationships between the dimensional attributes.
Add Attributes	Select this option to add a new attribute to the OLAP Dimension.
Add Attribute relationships	Select this option to add an attribute relationship for the selected OLAP Dimension.
Hierarchies	Select this option to perform one of the following: • **Add Hierarchy** - Select to add hierarchies. • **List Hierarchies** - Select to list hierarchies. • **List Hierarchy Elements** - Select to list hierarchy elements. • **Copy Hierarchies from Source** - Select to copy all hierarchies from the source table to the destination table. Source hierarchies are copied to the destination table automatically during table creation, but this feature is useful if the source table has been updated since destination table was created.

© Copyright, WhereScape Software Ltd All rights reserved.

Menu Options	Description
	• **Add hierarchy Element** - Select to add a new hierarchy element.
Version Control	A version of an OLAP Dimension is a copy of the metadata definition of the OLAP Dimension at the time of the versioning. This version information can be used to create a new OLAP Dimension or can simply be left as a backup and reference point. Select this option to perform one of the following: • **New Version** - Enables you to create a new version of an OLAP Dimension. • **Build Application** - Enables you to build an application file for the OLAP Dimension.
Delete Dimension metadata	Deletes the meta data of the selected OLAP cube, including all related information. A version of the object's metadata is normally auto created (depends on the settings in **Tools > Options > Metadata Versioning**).
Drop Analysis Services object	Select this option to drop an object associated with the OLAP Dimension in Analysis Services.
Retrofit dimension	Select this option to retrofit the OLAP Dimension from Analysis Services.
Create (Alter) dimension	Select this option to create the OLAP Dimension and supporting objects in Analysis Services (including cube database and dsv) based on the definition in WhereScape RED. This option requires connection and cube database information populated in the OLAP Dimension Properties.
Create via Scheduler	Select this option to submit a create of the OLAP Dimension via the RED scheduler.
Update	Select this option to process the OLAP Dimension in Analysis Services interactively from the WhereScape RED interface. **Tip:** This option requires connection and cube database information populated in the OLAP Dimension Properties.
Process Dimension via a Scheduler	Select this option to generate a WhereScape RED scheduler job to process the OLAP Dimension in Analysis Services. **Tip:** This option requires connection and cube database information populated in the OLAP Dimension Properties.
Documentation	Select this option to generate (or read if already generated) the WhereScape RED HTML documentation for the selected object. Select this option to perform one of the following: • **Display** the documentation available for the selected object. Refer to **Reading the Documentation** for details. • **Create** the documentation for the selected object. Refer to **Creating Documentation** for details.
Projects	Select this option to perform one of the following: • **Remove from Project** - Select to remove the OLAP Dimension from a project. Refer to **Removing Objects from Projects** for details. • **Add to Project** - Select to add the OLAP Dimension to a project. Refer to **Adding Objects to Projects** for details. • **List Projects** - Displays a list of projects which contain the current object, results are shown in the bottom pane. Refer to **List Project Memberships for Objects** for details. **Note:** Multiple objects can be selected by double-clicking the Object Group icon in the left pane and then Ctrl + clicking multiple objects in the Drop Target (middle) pane.

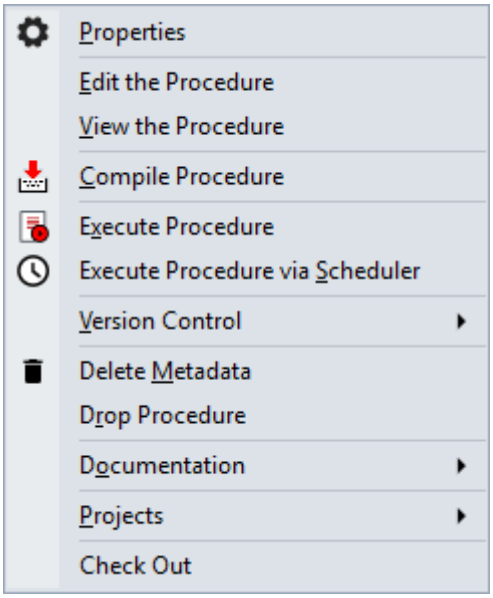
© Copyright, WhereScape Software Ltd All rights reserved.

Menu Options	Description
Check Out	Select this option to check out the object for editing and prevent any other users from being able to modify, update or delete any of their associated objects while you are making changes. Refer to **Object Check-Outs and Check-Ins** for details.
Impact	Select this option to produce a **Track Back** diagram on the OLAP cube dimension. Select this option to perform one of the following: • **Track Back Report** • **Track Forward Report** • **Track Back Diagram** • **Track Forward Diagram** • **Dependent Jobs Report**

Procedures

Procedures are commonly auto built through the **Properties** screen of one of the table types. They can also be created manually. Once created, they can be edited, compiled, etc.

Following is an example of the menu displayed when you right-click a procedure name.



If a procedure has been locked as a result of the WhereScape RED utility being killed or failing or a database failure then it can be unlocked via the **Properties** screen associated with the procedure.

Note:

WhereScape RED supports the use of keyboard shortcuts—the underlined letter of a menu option. For example, pressing **P** opens the **Properties** window of the selected table, pressing **E** displays the procedure for editing, etc.
Ensure that the **Windows > Control Panel > Ease of Access** setting associated with keyboard shortcuts is enabled, to display keyboard shortcuts in RED. Refer to the relevant MS Windows documentation for details.

Menu Options	Description
Edit the Procedure	Select this option to invoke the procedure editor and loads the procedure. A procedure can be compiled and executed within the procedure editor.
View the Procedure	Select this option to display a read only copy of the procedure. If the procedure is locked by another user then viewing the procedure is the only option available.
Compile Procedure	Select this option to compile the procedure from the metadata.

© Copyright, WhereScape Software Ltd All rights reserved.

Menu Options	Description
Execute Procedure	Select this option to execute the procedure and displays the results in the results window.
Execute Procedure via Scheduler	Select this option to set up a job to execute the procedure via the scheduler.
Version Control	A version of a Procedure can be created at any time via the Version Control, New Version menu option. The various versions of the procedure can be viewed from within procedure editor, or a new procedure can be created from the version. Select this option to perform one of the following: • **New Version** - Enables you to create a new version of the selected Procedure. • **Build Application** - Enables you to build an application file for the selected Procedure. • **Revert to Version** - Enables you to revert to a previous version of the selected Procedure. Displays a list of the available versions of the selected procedure, from which you can select and revert.
Delete metadata	Select this to option perform one of the following delete options: • **Delete metadata and drop object** - This option deletes the metadata definition for the procedure and drops the procedure object from the database (default). This is a permanent delete and no recovery is provided, please use with caution. • **Delete metadata only** - This option deletes the metadata definition for the procedure. A version of the object's metadata is normally auto created (depends on the settings in **Tools > Options > Metadata Versioning**).
Drop Procedure	Select this option to drop the procedure from the metadata and from the database as well. **Note:** Procedures cannot be dropped or deleted if they are in use or being edited with the Edit Lock set in the **Procedure Properties** window.
Documentation	Select this option to generate (or read if already generated) the WhereScape RED HTML documentation for the selected object. Select this option to perform one of the following: • **Display** the documentation available for the selected object. Refer to **Reading the Documentation** for details. • **Create** the documentation for the selected object. Refer to **Creating Documentation** for details.
Projects	Select this option to perform one of the following: • **Remove from Project** - Select to remove the procedure from a project. Refer to **Removing Objects from Projects** for details. • **Add to Project** - Select to add the procedure to a project. Refer to **Adding Objects to Projects** for details. • **List Projects** - Displays a list of projects which contain the current object, results are shown in the bottom pane. Refer to **List Project Memberships for Objects** for details. **Note:** Multiple objects can be selected by double-clicking the Object Group icon in the left pane and then Ctrl + clicking multiple objects in the Drop Target (middle) pane.

© Copyright, WhereScape Software Ltd All rights reserved.

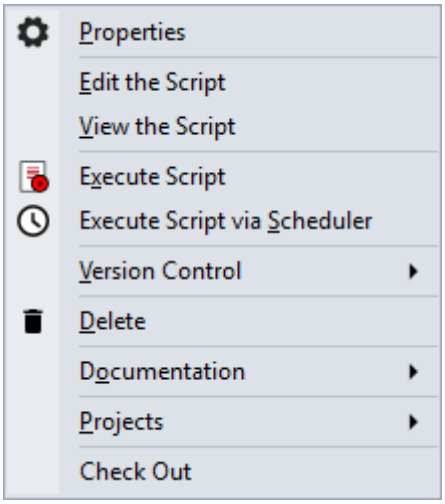
Menu Options	Description
Check Out	Select this option to check out the object for editing to prevent any other users from being able to modify, update or delete any of their associated objects while you are making changes. Refer to **Object Check-Outs and Check-Ins** for details.
Build	Select this option to build a Scheduler job or an application based on the Procedure table(s) selected from the middle pane. Refer to **Building Scheduler Jobs from Object Groups** and **Building Applications from Object Groups** for details. This option is available from the context menu of Procedure tables listed in the middle pane.

Scripts

Scripts are very similar in their operations to **Procedures**. Some key differences are as follows:

- If a script is deleted, it is only removed from the metadata. However, because WhereScape RED never stores the script on the host system, this removes the script permanently.
- Unix scripts, Windows CMD scripts and **Windows PowerShell Scripts** are available on Load objects.
- Windows CMD scripts and Windows PowerShell scripts are available on all table object types, where the table is stored in a custom database target and when using a Windows scheduler.
- The table object needs to specify the connection appropriate for the script (typically Windows or Unix).

Following is an example of the menu displayed when you right-click a script object.



Note:

WhereScape RED supports the use of keyboard shortcuts—the underlined letter of a menu option. For example, pressing **P** opens the **Properties** window of the selected table, pressing **D** deletes the script metadata, etc.
Ensure that the **Windows > Control Panel > Ease of Access** setting associated with keyboard shortcuts is enabled, to display keyboard shortcuts in RED. Refer to the relevant MS Windows documentation for details.

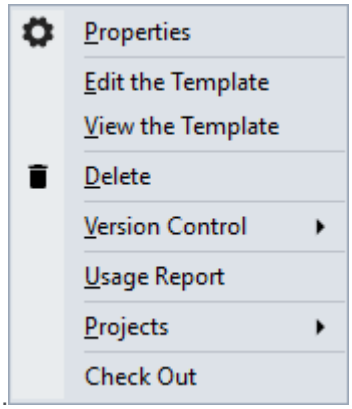
© Copyright, WhereScape Software Ltd All rights reserved.

Templates

Template objects are used to generate the update routines of table objects in RED, e.g. generation of DDL, update/custom procedures and host scripts. **Templates** are very similar in their operations to **Scripts**.

The additional context menu option is **Usage Report** which enables you identify the table objects that are associated with a selected Template object.

Following is an example of the menu displayed when you right-click a Template object.



Note:

WhereScape RED supports the use of keyboard shortcuts—the underlined letter of a menu option. For example, pressing **P** opens the **Properties** window of the selected template, pressing **U** displays the RED objects that are using the template, etc.
Ensure that the **Windows > Control Panel > Ease of Access** setting associated with keyboard shortcuts is enabled, to display keyboard shortcuts in RED. Refer to the relevant MS Windows documentation for details.

Selecting **Usage Report** displays a list of objects that are currently using the selected template in the middle pane.

Template (wsl_sqlserver_proc_dv_perm) Usage Report

Name	Type	Template Usage
src_h_customer_DV_1	Source Mapping	Update Procedure
src_h_customer_DV_2	Source Mapping	Update Procedure
dim_customer	Dimension	Custom Procedure
l_customer_DV	Link	Update Procedure
l_order_header_DV	Link	Update Procedure
s_customer_DV	Satellite	Update Procedure
s_order_header_DV	Satellite	Update Procedure
h_customer_DV	Hub	Update Procedure
h_customer_code_merge	Hub	Update Procedure
ds_product	Data Store	Update Procedure
ds_order_line	Data Store	Update Procedure
ds_order_header	Data Store	Update Procedure
h_product_DV	Hub	Update Procedure
l_product_DV	Link	Update Procedure
s_product_DV	Satellite	Update Procedure

You can select multiple objects from the list and then select **Code** from the right-click context menu to perform a bulk update procedure or script of the selected objects. Refer to **Templates** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

You can also define default templates to use for the update routines of table objects in RED by Connection type—refer to **Routine Templates** for details.

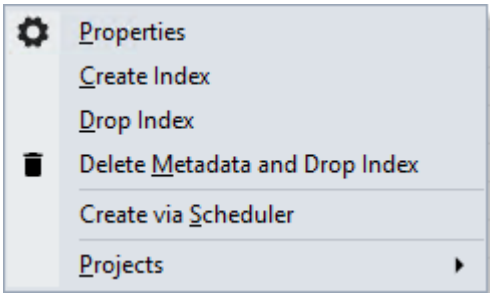
Note:

The **Version Control** sub menu for Template objects provide **New Version**, **Build Application** and **Duplicate Object** options.

Indexes

Indexes are always associated with a table. To define a new index, the menu option associated with the table that is to have the index must be used.

Following is an example of the context menu displayed when you right-click an index.



Note:

WhereScape RED supports the use of keyboard shortcuts—the underlined letter of a menu option. For example, pressing **P** opens the **Properties** window of the selected table, pressing **C** creates the index, etc. Ensure that the **Windows > Control Panel > Ease of Access** setting associated with keyboard shortcuts is enabled, to display keyboard shortcuts in RED. Refer to the relevant MS Windows documentation for details.

If a procedure has been locked as a result of the WhereScape RED utility being killed or failing or a database failure then it can be unlocked via the Properties screen associated with the procedure.

Menu Options	Description
Properties	Select this option to display the **Properties** screen, which contains the entire definition of the index, including the columns in use and the index type etc. The way that the scheduler handles the index is also defined in the Properties. An index can be set so that it is dropped by the scheduler prior to a table update and then rebuilt by the scheduler, once the update has been completed. It can also be defined for rebuild on certain days.
Create Index	Select this option to create an index in the database. **Tip:** It may take some time for large indexes, and in such cases, it is better to schedule a create of the index. Refer to **Scheduler** for details. This menu option is intended for use when working with prototype data volumes.
Drop Index	Select this option to drop the index in the database.
Delete Metadata and Drop Index	Select this option to remove metadata definition of the index and drops it from the database. No recovery is possible once this option is executed. A version of the object's metadata is normally auto created (depends on the settings in **Tools > Options > Metadata Versioning**).
Create via Scheduler	Select this option to submit a create of the index to the scheduler.
Projects	Select this option to: • **Remove from Project** - Select to remove the index from a project. Refer to **Removing Objects from Projects** for details.

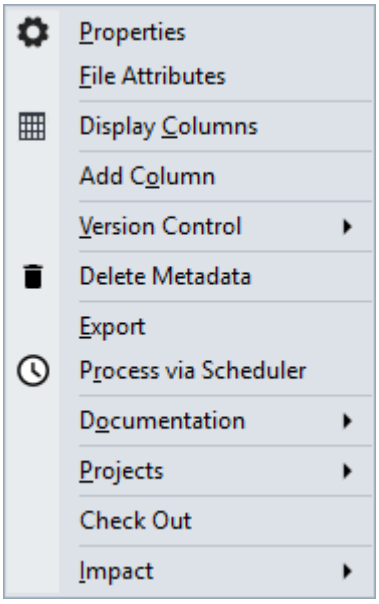
© Copyright, WhereScape Software Ltd All rights reserved.

Menu Options	Description
	• **Add to Project** - Select to add the index to a project. Refer to **Adding Objects to Projects** for details. • **List Projects** - Displays a list of projects which contain the current object, results are shown in the bottom pane. Refer to **List Project Memberships for Objects** for details. **Note:** Multiple objects can be selected by double-clicking the Object Group icon in the left pane and then Ctrl + clicking multiple objects in the Drop Target (middle) pane.
Build	Select this option to build a Scheduler job or an application based on the Index table(s) selected from the middle pane. Refer to **Building Scheduler Jobs from Object Groups** and **Building Applications from Object Groups** for details. This option is available from the context menu of Index tables listed in the middle pane.

Exports

Export objects are used in WhereScape RED to produce ASCII files from a single database Table or View for a downstream feed. Some or all of the columns in a Table or View can be exported

Following is an example of the context menu displayed when you right-click an Export object.



Note:

WhereScape RED supports the use of keyboard shortcuts—the underlined letter of a menu option. For example, pressing **P** opens the **Properties** window of the selected table, pressing **C** creates the index, etc. Ensure that the **Windows > Control Panel > Ease of Access** setting associated with keyboard shortcuts is enabled, to display keyboard shortcuts in RED. Refer to the relevant MS Windows documentation for details.

Tip:

All the other context menu options for Export objects are described under the other object types.

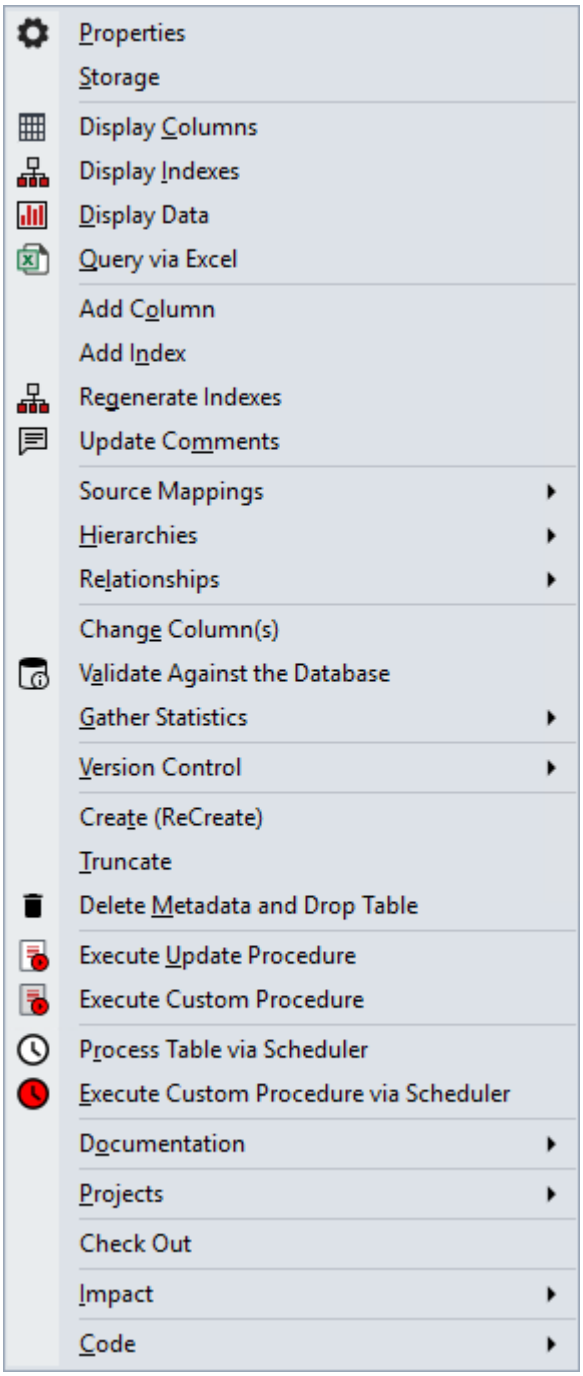
Menu Options	Description
File Attributes	Select this option to display the **Properties** window for the Export table, focusing on the **File Attributes** tab within this window.

© Copyright, WhereScape Software Ltd All rights reserved.

Hub Tables

Hub table objects in WhereScape RED contain hash key(s), business key(s), load date and record source. This table can be created via a Wizard—refer to **Creating the Hub, Link and Satellite Tables** for details.

Following is an example of the menu displayed when you right-click a Hub table object.



Tip:

The bulk of the menu options are same as for the other table objects described in this section.

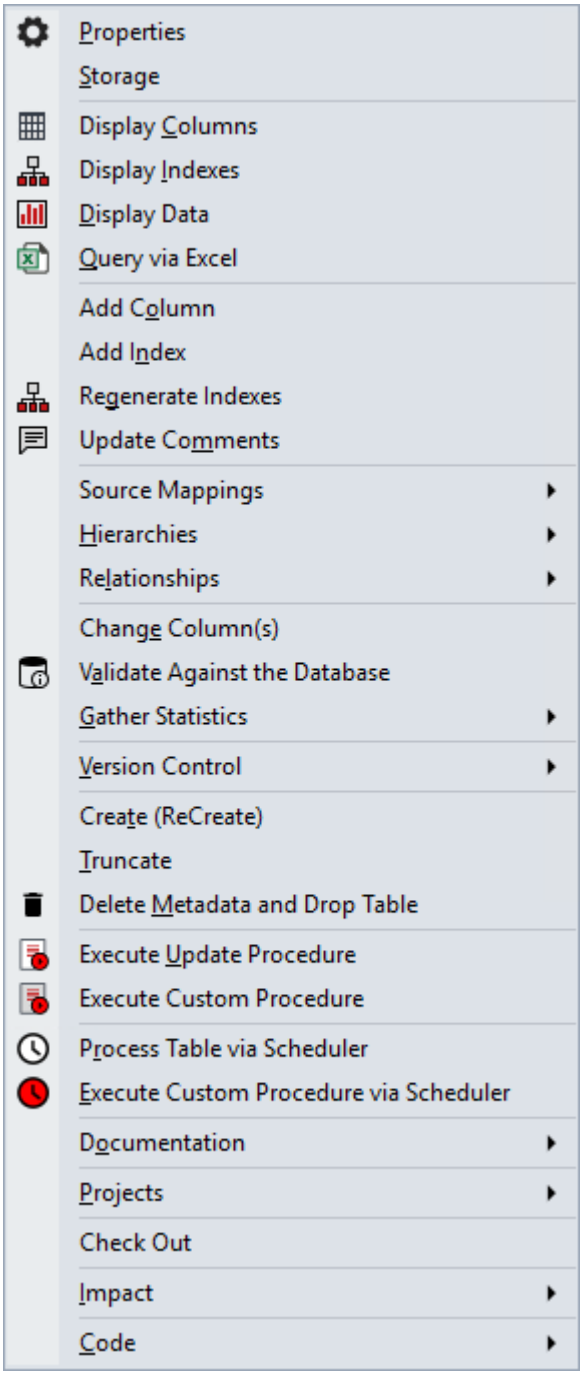
© Copyright, WhereScape Software Ltd All rights reserved.

Link Tables

Link table objects are many-to-many tables representing current and past relationships between two or more Hub entities and are used to describe associations, transactions, hierarchies and redefinitions of Hub entities in a Data Vault. Links have their own hash key and the hash keys for the Hubs that are linked, as well as a Load Date and Record Source. The attributes describing the context of a link are stored in Satellite tables.

This table can be created via a Wizard—refer to **Creating the Hub, Link and Satellite Tables** for details.

Following is an example of the menu displayed when you right-click a Link table object.



Tip:

The bulk of the menu options are same as for the other table objects described in this section.

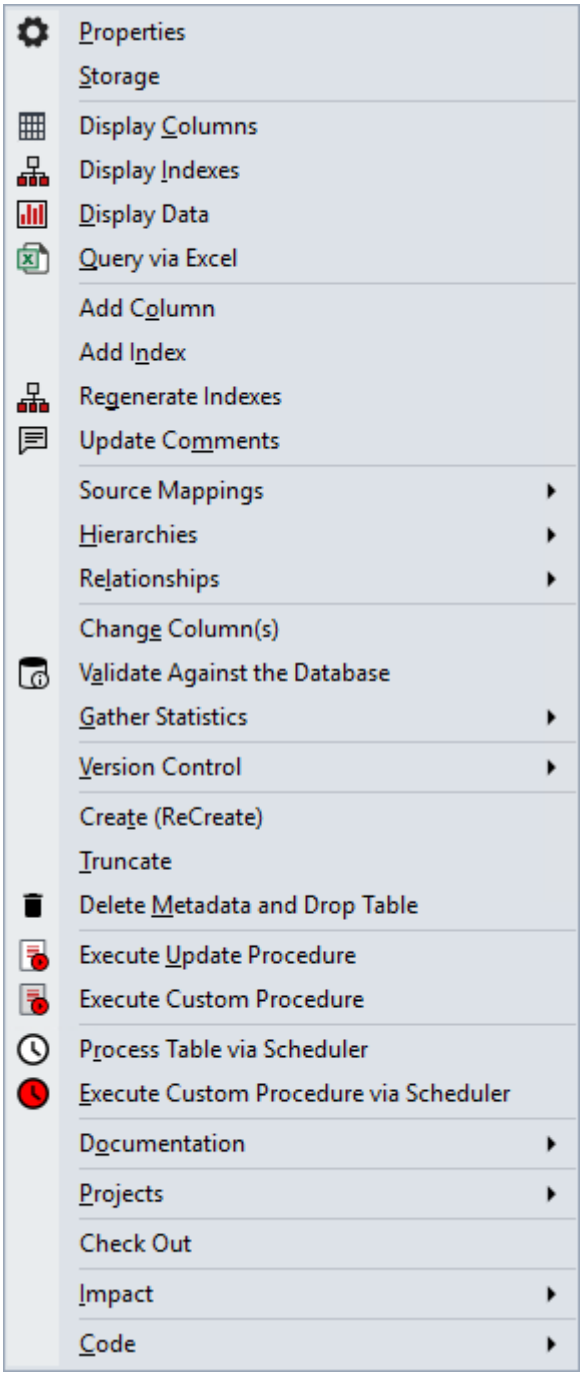
© Copyright, WhereScape Software Ltd All rights reserved.

Satellite Tables

Satellite tables are Data Vault objects that contain metadata which provides context for Hub and Link entities at a given time or over a period of time. Each Satellite entity can contain information on one Hub or Link. Satellite tables contain a hash key for the parent Hub or Link, a timestamp for the date of change and relevant descriptive fields.
Satellites are usually created once per source system. Because descriptive attributes can change at different rates, Satellites can also be created based on rate of change.

This table can be created via a Wizard—refer to **Creating the Hub, Link and Satellite Tables** for details.

Following is an example of the menu displayed when you right-click a Satellite table object.



Tip:

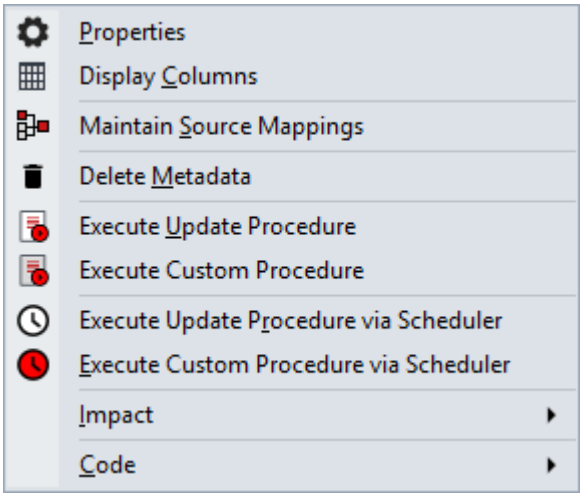
The bulk of the menu options are same as for the other table objects described in this section.

© Copyright, WhereScape Software Ltd All rights reserved.

Source Mapping

Source Mapping objects are used in WhereScape RED to define source column mappings, when there is a requirement for the target table to be populated by more than one set of source tables. These objects support the generation of an update procedure for the target table, sourced from each set of the source tables. Refer to **Multi Source Processing** for details.

Following is an example of the menu displayed when you right-click a source mapping object.



Note:

WhereScape RED supports the use of keyboard shortcuts—the underlined letter of a menu option. For example, pressing **P** opens the **Properties** window of the selected table, pressing **C** display columns, etc. Ensure that the **Windows > Control Panel > Ease of Access** setting associated with keyboard shortcuts is enabled, to display keyboard shortcuts in RED. Refer to the relevant MS Windows documentation for details.

If a procedure has been locked as a result of the WhereScape RED utility being killed or failing or a database failure then it can be unlocked via the Properties screen associated with the procedure.

Menu Options	Description
Maintain Source Mappings	Select this option to launch the Source Mapping tool which enables you to manage the column mappings of the selected target table.
Execute Update Procedure via Scheduler	Select this option to set up a WhereScape RED scheduler job to execute the source mapping object's update procedure.
Execute Custom Procedure via Scheduler	Select this option to execute the procedure defined as the 'custom procedure' for the table via the WhereScape RED scheduler.
Impact	Select this option to perform one of the following: • **Track Back Report** – Lists the source tables of the selected source mapping object. • **Track Forward Report** – Lists the parent table object of the selected source mapping object. • **Dependent Jobs Report** – Lists jobs that includes the currently selected source mapping object or the parent table object.
Code	Select this option to view a procedure attached to a table or to rebuild or regenerate the table's update procedure.

Tip:

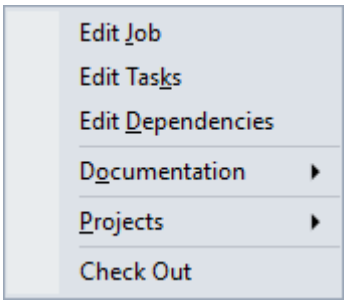
All the other options for source mappings are described under the other object types.

© Copyright, WhereScape Software Ltd All rights reserved.

Scheduler Jobs

Jobs, such as data loads and updates can be run in background mode and/or at a pre-determined time, using the RED scheduler. Refer to **Scheduler** for details.

Following is an example of the menu displayed when you right-click a scheduler job object.



Note:

WhereScape RED supports the use of keyboard shortcuts—the underlined letter of a menu option. For example, pressing **P** opens the **Properties** window of the selected table, pressing **C** creates the index, etc. Ensure that the **Windows > Control Panel > Ease of Access** setting associated with keyboard shortcuts is enabled, to display keyboard shortcuts in RED. Refer to the relevant MS Windows documentation for details.

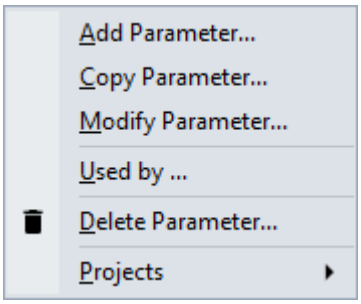
Menu Options	Description
Edit Job	Select this option to launch the **Job Definition** window which enables you to edit the job parameters and settings of the selected scheduler job. Refer to **Editing a Job** for details.
Edit Tasks	Select this option to launch the **Define tasks** window which enables you to edit the tasks included in the selected scheduler job. Refer to **Editing Tasks in a Job** for details.
Edit Dependencies	Select this option to launch the **Dependencies** window which enables you to edit the dependencies between the tasks included in the selected scheduler job. Refer to **Editing Task Dependencies** for details.
Documentation	Select this option to: • **Display** the documentation available for the selected scheduler job. Refer to **Reading the Documentation** for details. • **Create** the documentation for the selected scheduler job. Refer to **Creating Documentation** for details.
Projects	Select this option to perform one of the following: • **Remove from Project** - Select to remove the Scheduler job from a project. Refer to **Removing Objects from Projects** for details. • **Add to Project** - Select to add the Scheduler job to a project. Refer to **Adding Objects to Projects** for details. • **List Projects** - Displays a list of projects which contain the current object, results are shown in the bottom pane. Refer to **List Project Memberships for Objects** for details. **Note:** Multiple objects can be selected by double-clicking the Object Group icon in the left pane and then Ctrl + clicking multiple objects in the Drop Target (middle) pane.
Check Out	Select this option to check out the object for editing to prevent any other users from being able to modify, update or delete any of their associated objects while you are making changes. Refer to **Object Check-Outs and Check-Ins** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

Parameters

Parameters are a means of passing information between two or more procedures and between the WhereScape RED environment and procedures. Parameters can be added or removed from a Project.

Following is an example of the menu displayed when you right-click a parameter object.



Note:

WhereScape RED supports the use of keyboard shortcuts—the underlined letter of a menu option. For example, pressing **P** opens the **Properties** window of the selected table, pressing **C** creates the index, etc. Ensure that the **Windows > Control Panel > Ease of Access** setting associated with keyboard shortcuts is enabled, to display keyboard shortcuts in RED. Refer to the relevant MS Windows documentation for details.

Menu Options	Description
Add Parameter	Select this option to launch the **Parameter Maintenance** window which enables you to define a parameter and its value.
Copy Parameter	Select this option to launch the **Parameter Maintenance** window which enables you to copy the selected parameter which you can use in other objects.
Modify Parameter	Select this option to launch the **Parameter Maintenance** window which enables you to modify the selected parameter and its value.
Used by	Select this option to list the objects that are using the selected parameter in the **Results** pane.
Delete Parameter	Select this option to delete the selected parameter.
Projects	Select this option to perform one of the following: • **Remove from Project** - Select to remove the parameter from a project. Refer to **Removing Objects from Projects** for details. • **Add to Project** - Select to add the parameter to a project. Refer to **Adding Objects to Projects** for details. • **List Projects** - Displays a list of projects which contain the current object, results are shown in the bottom pane. Refer to **List Project Memberships for Objects** for details. **Note:** Multiple objects can be selected by double-clicking the Object Group icon in the left pane and then Ctrl + clicking multiple objects in the Drop Target (middle) pane.

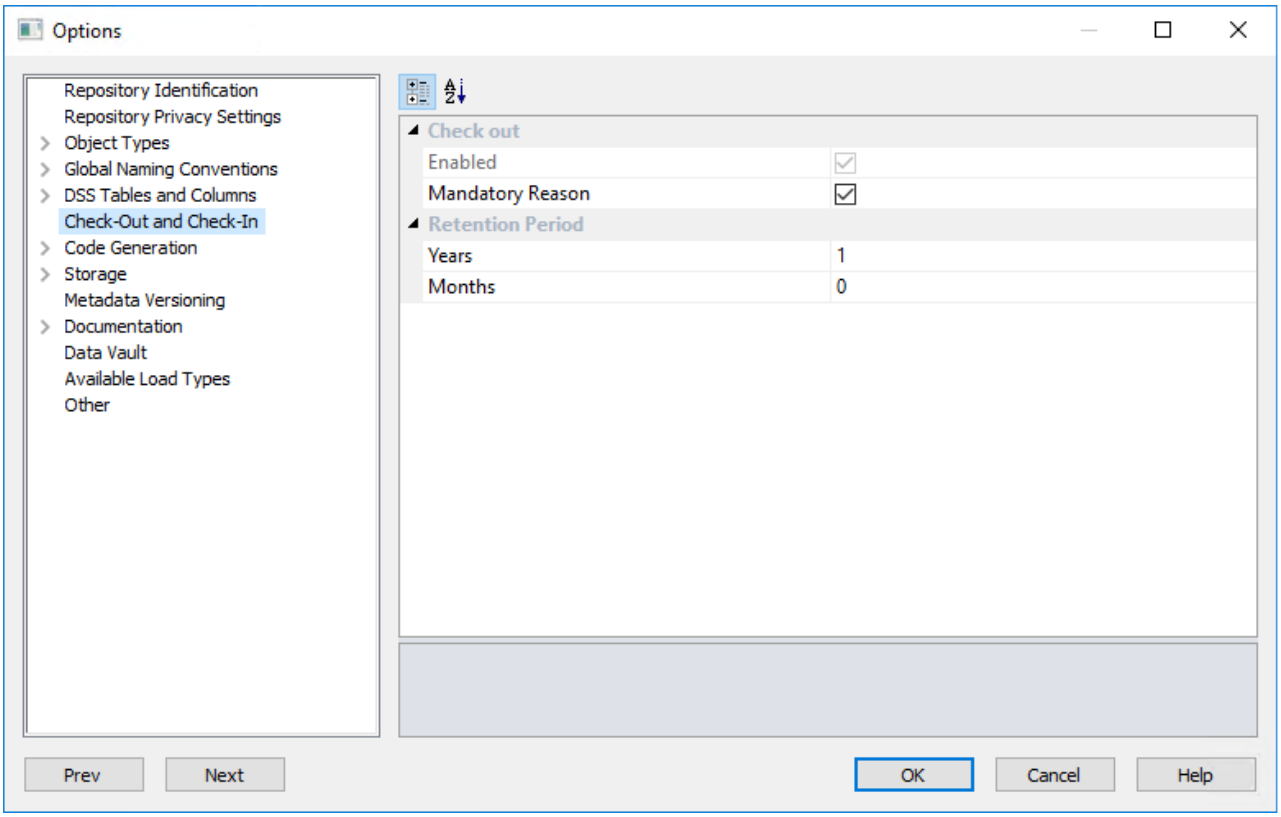
© Copyright, WhereScape Software Ltd All rights reserved.

Object Check-Outs and Check-Ins

Objects in WhereScape RED can be checked out for editing to prevent other users from being able to modify, update or delete any of their associated objects, while you are making changes to them.

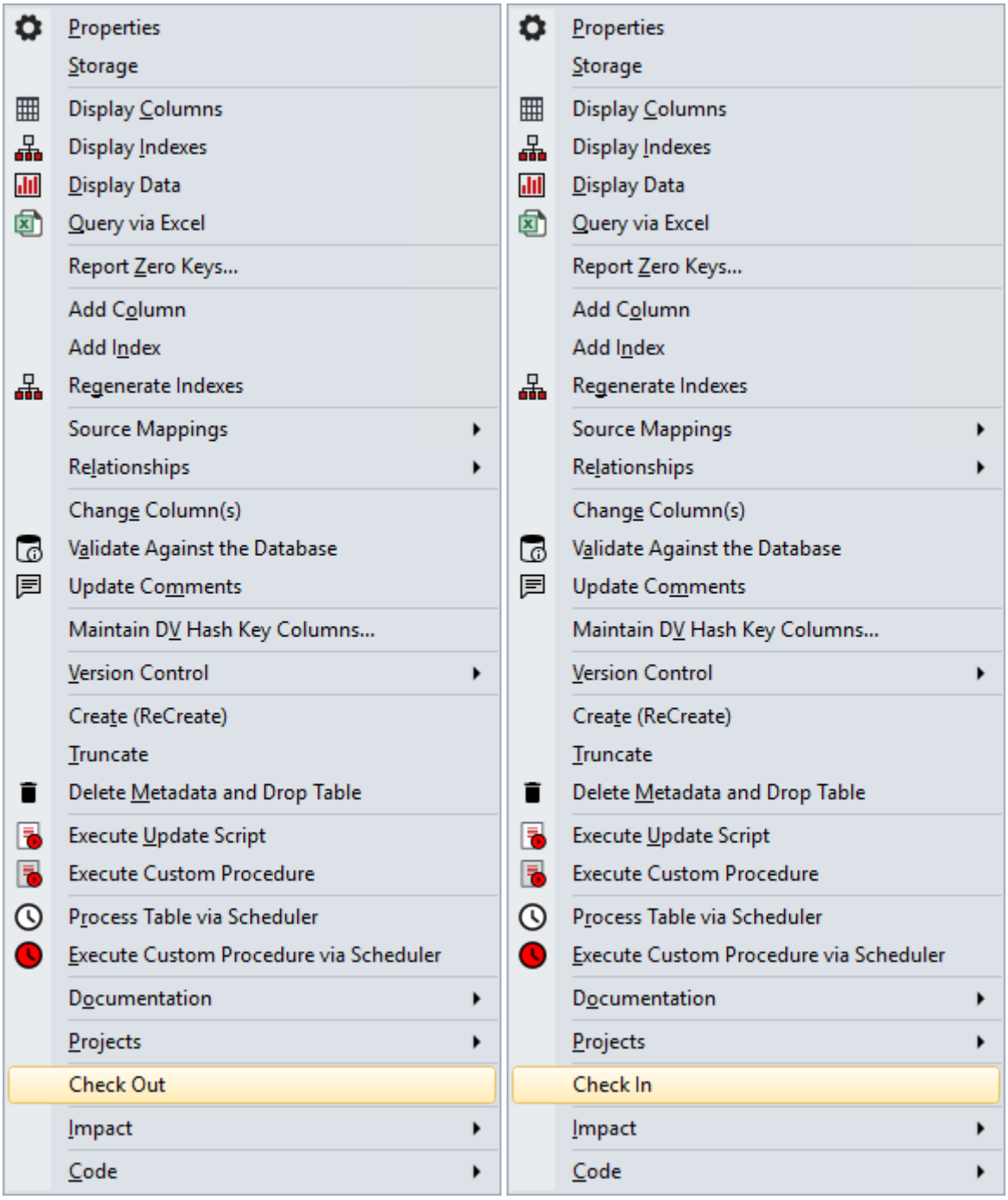
The Check-Outs and Check-Ins functionality is configured from **Tools > Options > Check-Out and Check-In** window.

The Check-Out option is enabled by default, as shown below:



© Copyright, WhereScape Software Ltd All rights reserved.

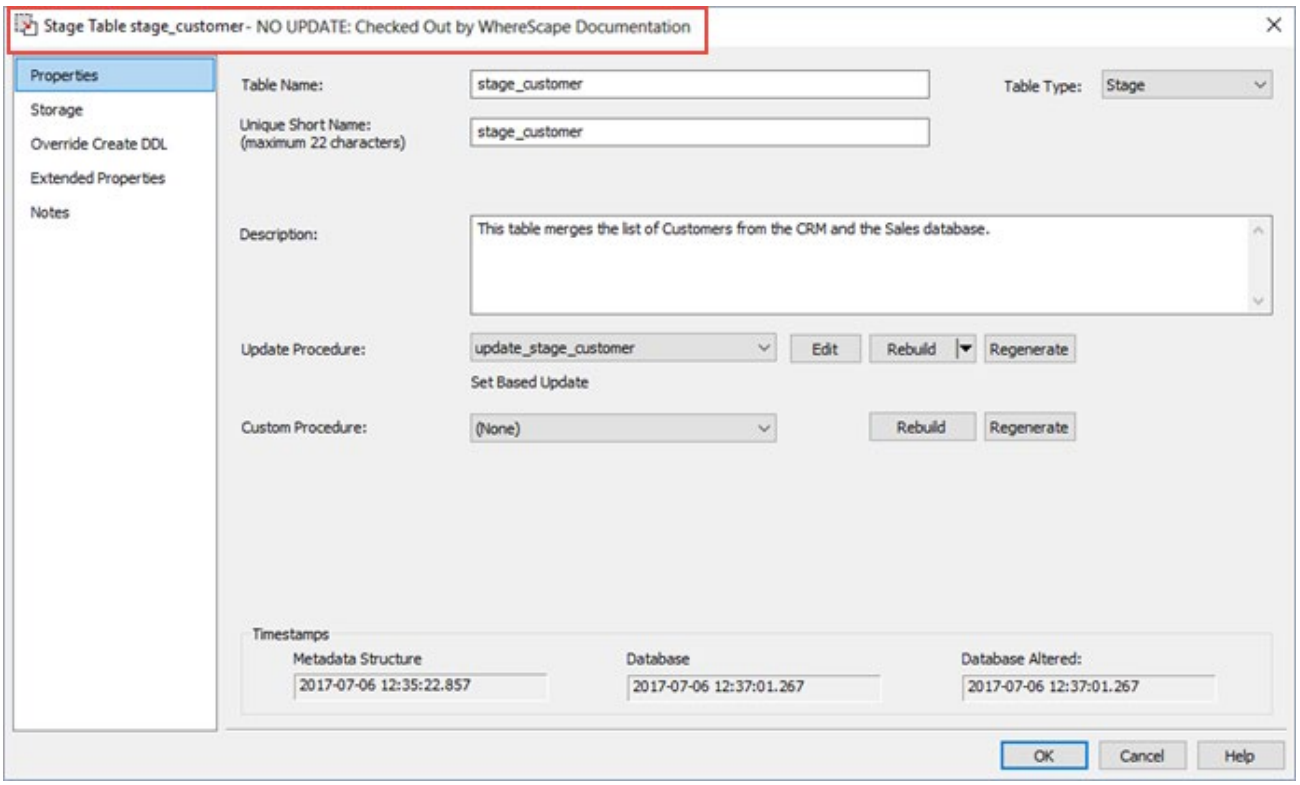
The following screens displays the setting to check-out/check-in objects in WhereScape RED through the object's right-click context menus.



To check-in an object, simply right-click it and select **Check In** from the context menu.

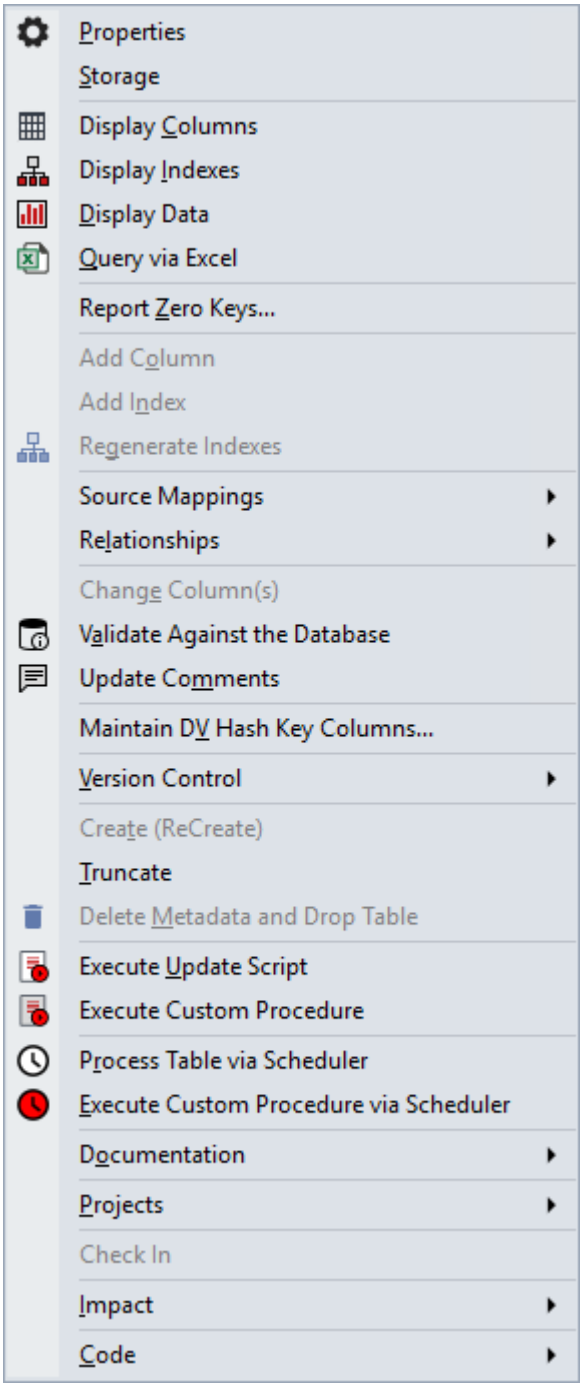
© Copyright, WhereScape Software Ltd All rights reserved.

Checked out objects and their associated procedures and scripts cannot be modified, updated or deleted by other users. If a different user tries to access the **Properties** or **Procedure** screens for those objects, the fields will not be available and headed NO UPDATE: Checked Out by (user name).



© Copyright, WhereScape Software Ltd All rights reserved.

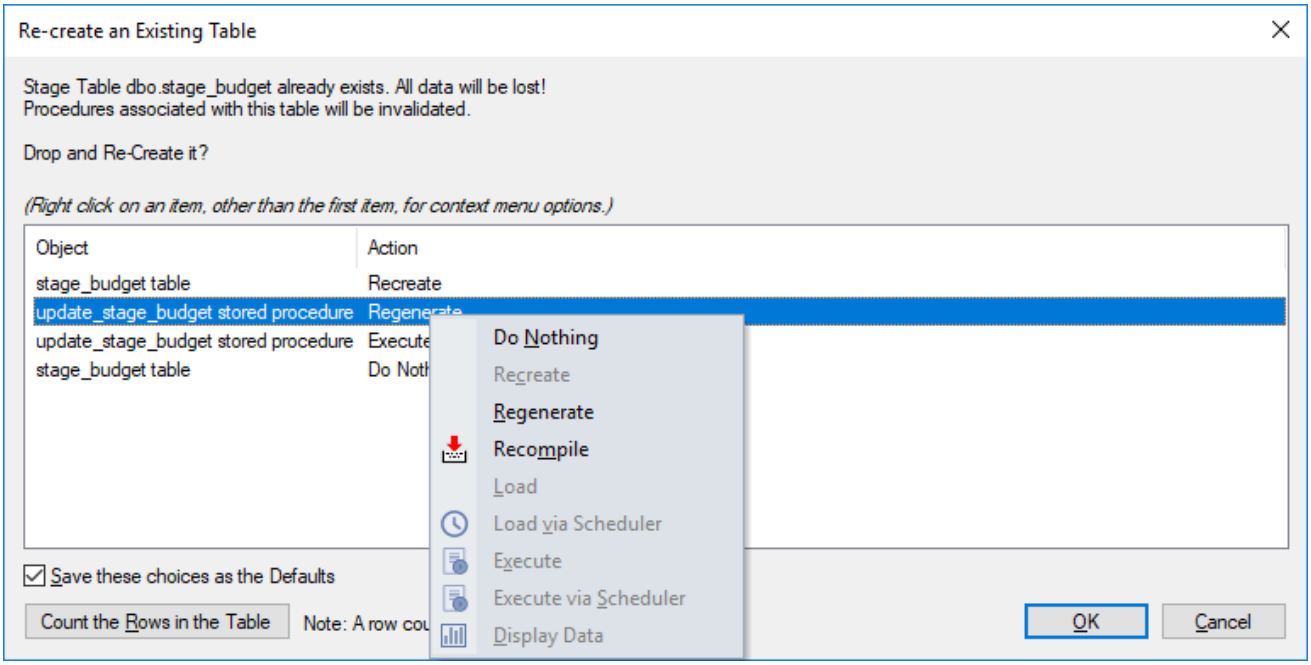
Other users logging into the same metadata repository will have some of the checked-out object's context menu options unavailable as shown below:



© Copyright, WhereScape Software Ltd All rights reserved.

Re-create an Existing Table

The Re-create window, as shown in the following screen provides available options for dropping and recreating an existing table and associated procedures. Objects and actions available depend on the table being recreated.



Actions in the table are completed from top to bottom; the action performed on each object can be set as desired:

Object(s)	Action	Description
Load Tables	Do Nothing (Default)	Do nothing for this object, skip to the next action.
	Load	Perform an interactive load of the data into the table. WhereScape recommends performing this via the Scheduler for large tables.
	Load via Scheduler	Add a data load task to the scheduler. Useful for large tables where processing may take some time.
Load Script	Regenerate	Regenerate the table's load script. This option is only available for Load tables with a load type of script and with the source connection setting defined. If the Load table does not have an existing script then a new script is created.
Update procedure	Do Nothing (Default)	Do nothing for this object, skip to the next action.
	Execute	Execute the update procedure. WhereScape recommends performing this via the Scheduler for large tables.
	Execute via Scheduler	Execute the update procedure via the scheduler. Useful for large tables where processing may take some time.
Tables	Do Nothing	Do not display the data.
	Display Data	Display the table data after recreating the table and update procedure (if applicable). • This is the default setting for Views and Dimension Views. • This is the default setting for tables when an update procedure is present and has been executed in the previous action. • Unavailable when an update procedure is present but has not been executed in the Re-create window.

© Copyright, WhereScape Software Ltd All rights reserved.

Save these choices as the Defaults

This option can be used to store the current actions as the default choices for all object types. The Re-create window is then automatically populated with the saved actions each time it appears.

Count the Rows in the Table

Displays the total number of rows in the table at the bottom of the Re-create window. Knowing the number of rows in a table can help estimate how long Load and Execute actions may take.

© Copyright, WhereScape Software Ltd All rights reserved.

Organizing Objects

As described in the previous section, there are many types of object in WhereScape RED .

The objects in the metadata repository are displayed in the left pane of the Builder window. They are displayed in a tree structure which can be expanded and closed as required.

The tree can be refreshed by using **F5** or the **Ctrl+R** key.

Object Groups

The objects created in WhereScape RED are grouped together into **object groups** based on object type.

For example, we store all the dimension objects in the **Dimension** object group. Optionally, we can choose to display dimension tables as the Dimension object group and dimension views as the Dimension View object group.

Projects

These object groups are in turn stored within **Projects**. When WhereScape RED is first started, the special project called **All Objects** is the only project. This project will always contain all the objects that exist in the metadata repository.

Additional projects can be created if required. These additional projects can hold some or all of the objects, as seen in **All Objects**. An object as such, only exists once in the metadata. Therefore, if we have a Dimension object called dim_product, there is and can only be one copy of the object dim_product within the metadata.

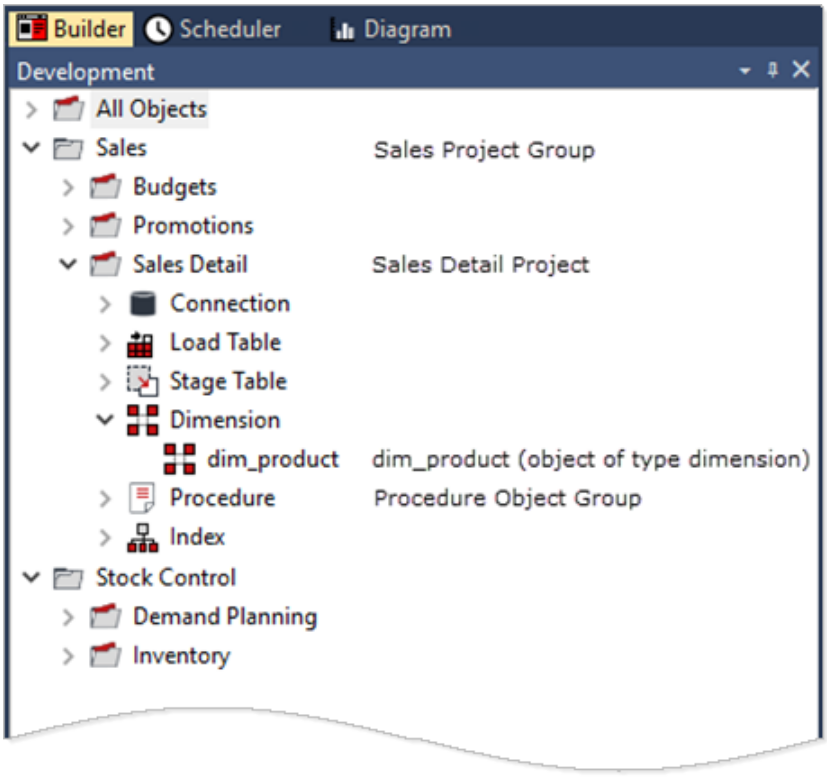
Projects are used to hold a group of objects that relate back to a similar module or analysis area of the data warehouse.

In the example below, we have additional projects called Budgets, Promotions, Sales Detail, Demand Planning and Inventory. In this way, projects enable us to restrict the amount of information (objects) we need to deal with to just those relevant to the area being worked on.

Projects can also be used to group objects for an upcoming code promote.

© Copyright, WhereScape Software Ltd All rights reserved.

The example below shows a metadata repository using two project groups (Sales and Stock Control) and five additional projects (Budgets, Promotions, Sales Detail, Demand Planning and Inventory).



Notes:

If you delete an object using the right-click context menu, the object is deleted from the metadata and removed from all Projects. You can remove the object from the project instead of deleting it. Use the object's right-click context menu to see the Projects to which the object is currently included. Refer to **List Project Memberships for Objects** for details.

It is important to understand that these projects are only a means of visualizing the objects. Even though an object may appear in many projects, it only exists once in the metadata.

To create a project, right-click a Project or Group listed in the left pane and select **New Project** from the context menu.

The **File > New Project** menu option can also be used. Projects can be renamed, removed from the project Group or deleted, using the right-click context menu of a selected project.

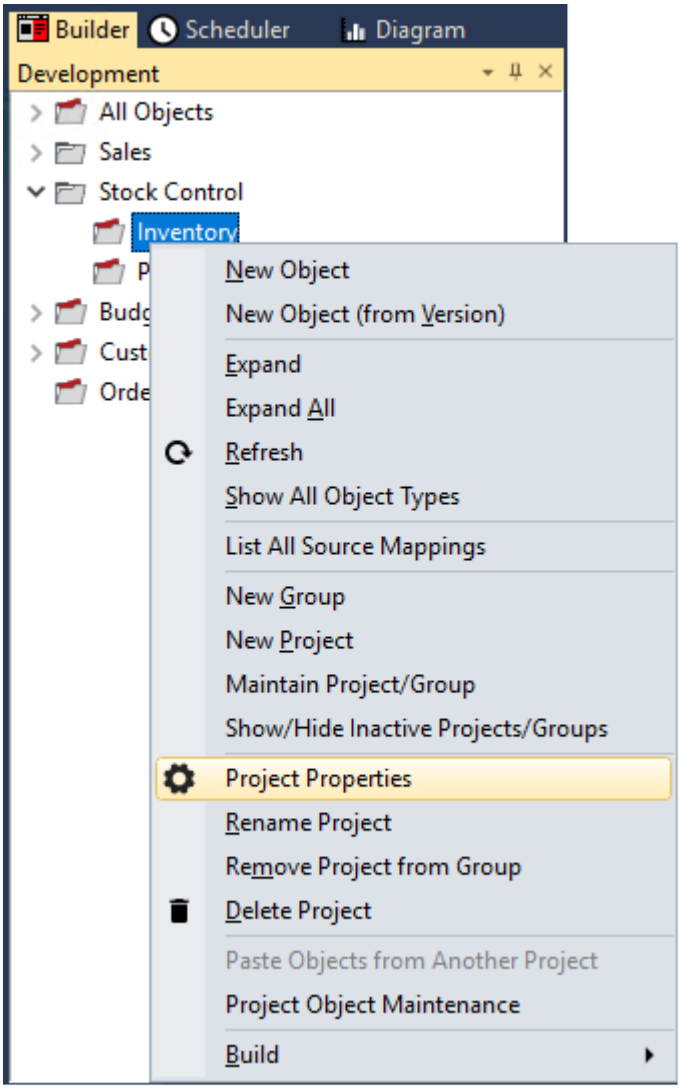
Tip:

You can automatically create a Project from a Diagram RED—refer to **Creating a Project from a Diagram** for details.

Deleting a Project does not delete the objects associated with it from the metadata; it simply removes their reference in the Project being deleted.

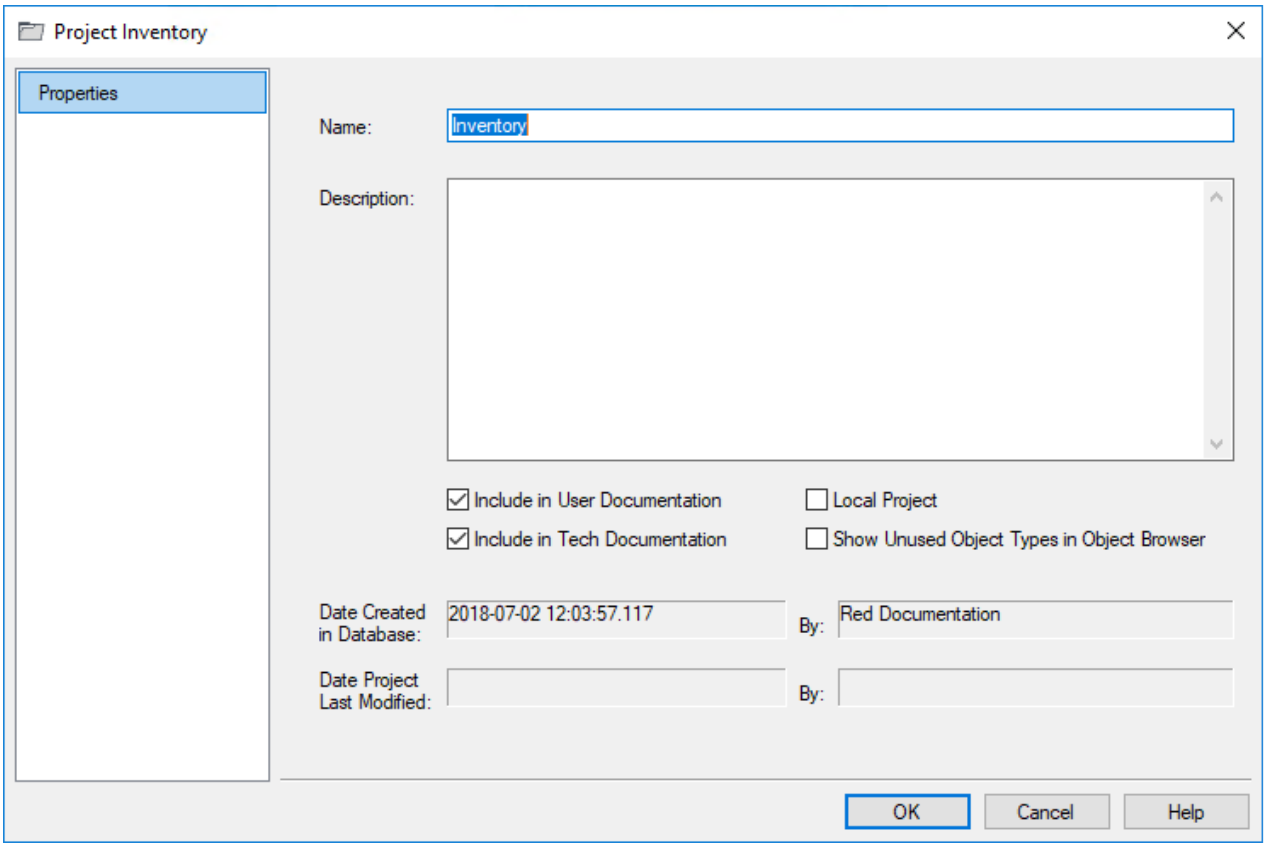
© Copyright, WhereScape Software Ltd All rights reserved.

To view or make changes to the project's Properties, right-click the Project name from the left pane and then select **Project Properties**.



© Copyright, WhereScape Software Ltd All rights reserved.

The Project Properties window is displayed:



Option	Description
Include in User Documentation	Select to include the project in the User Documentation. This is selected by default.
Include in Tech Documentation	Select to include in the Technical Documentation. This is selected by default.
Local Project	Select to exclude the local project in the Application files.
Show Unused Object Types in Object Browser	Select to display all object types in the Object Browser Pane. If not selected, only object types currently in use are displayed. Refreshing the Object Browser Pane is required after changing this setting.

© Copyright, WhereScape Software Ltd All rights reserved.

Project Groups

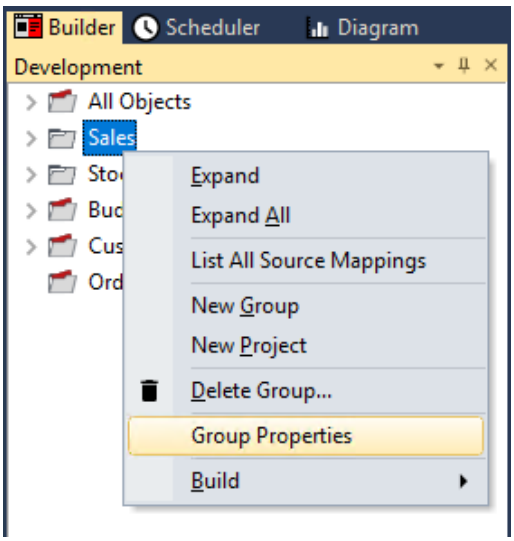
Projects can in turn be organized into **Groups**.

To create a Group, right-click a Project or Group listed in the left pane and select **New Group** from the context menu.

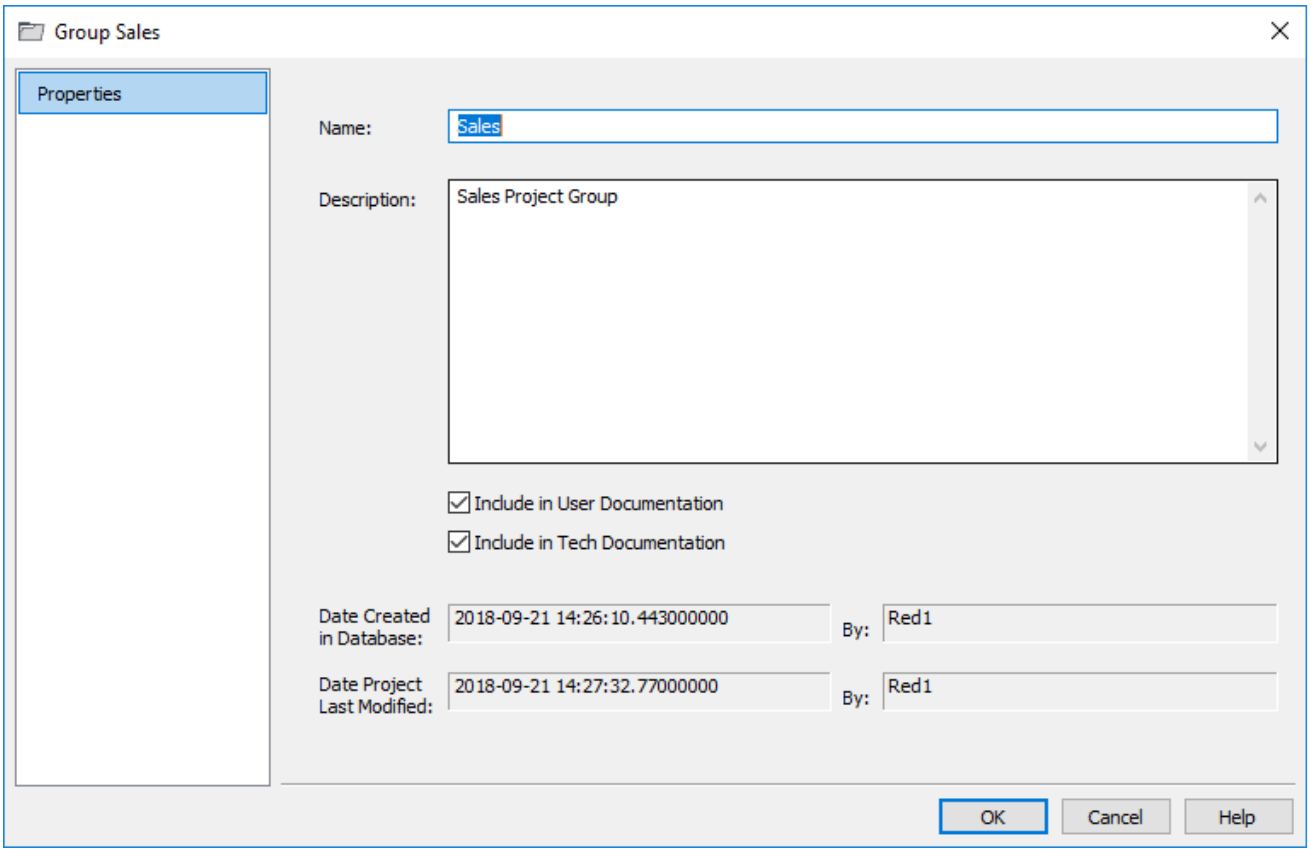
The **File > New Group** menu option can also be used. A project Group can be deleted using the right-click context menu of a selected group.

Deleting a Group does not delete its associated projects or objects from the metadata; it simply removes their reference in the Group being deleted. The projects of a deleted Group remain in the left pane.

To view or make changes to the group's Properties, right-click the Group name from the left pane and select **Group Properties**.



The Group Properties window is displayed:



© Copyright, WhereScape Software Ltd All rights reserved.

Adding Objects to Projects

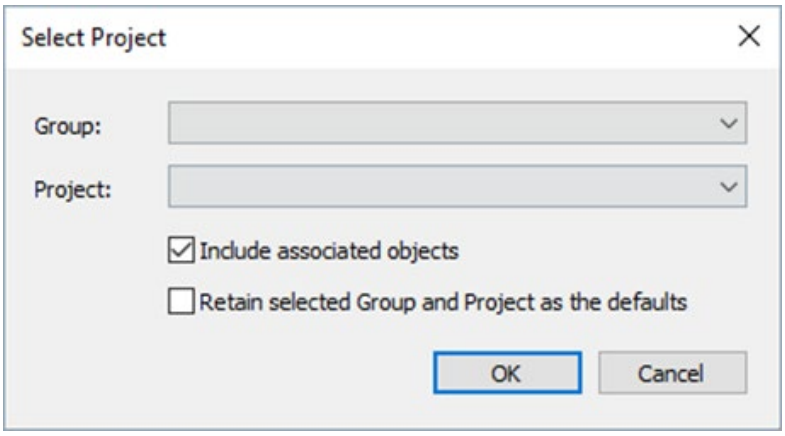
There are several ways to add objects to a project:

- Click an object in the left pane, press **Ctrl+C**, click the target project (either the project folder or any object group or object within it) and press **Ctrl+V**.
- **Drag** the object to the required project.
- Right-click an object in the left pane and select **Projects > Add to Project**.
- Highlight a number of objects in the middle pane, right-click and select **Projects > Add to Project**.
- **Use the Project Object Maintenance Facility** (refer to Using Project Object Maintenance for details).

Adding an Object to an Independent Project (a project that is not in a group)

To add an object to an independent project:

1. Right-click an object from the **Objects** list pane and select **Projects > Add to Project**.
   The **Select Project** window is displayed:

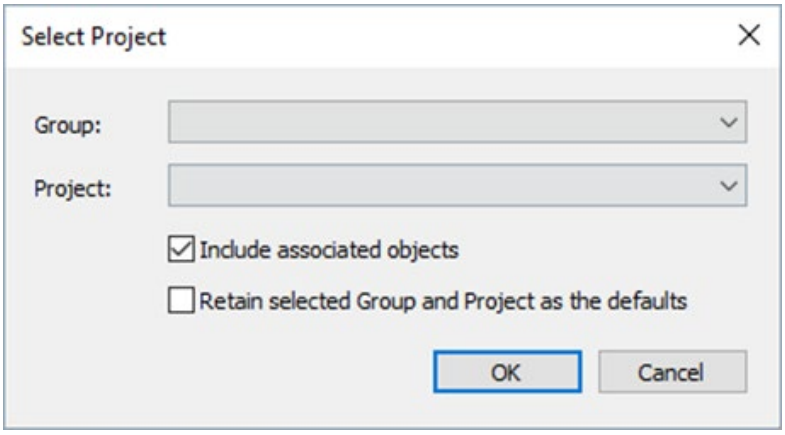


2. Select a project from the **Project** drop-down list and click **OK**.

Adding an Object to a Project in a Group

To add an object to a project that belongs to a group:

1. Right-click on an object from the **Objects** list pane and select **Projects > Add to Project**.
   The **Select Project** window is displayed:



2. Select a group from the **Group** drop-down list.
3. Select a project from the **Project** drop-down list and click **OK**.

Notes:

- Selecting the **Include associated objects** check box, also adds any indexes, procedures and scripts to the selected project.

© Copyright, WhereScape Software Ltd All rights reserved.

Notes:

- Selecting the **Retain selected Group and Project as the defaults** check box enables you to retain the last values defined in the Group and Project fields and use them by default when an object is added to a Project or Group.

Removing Objects from Projects

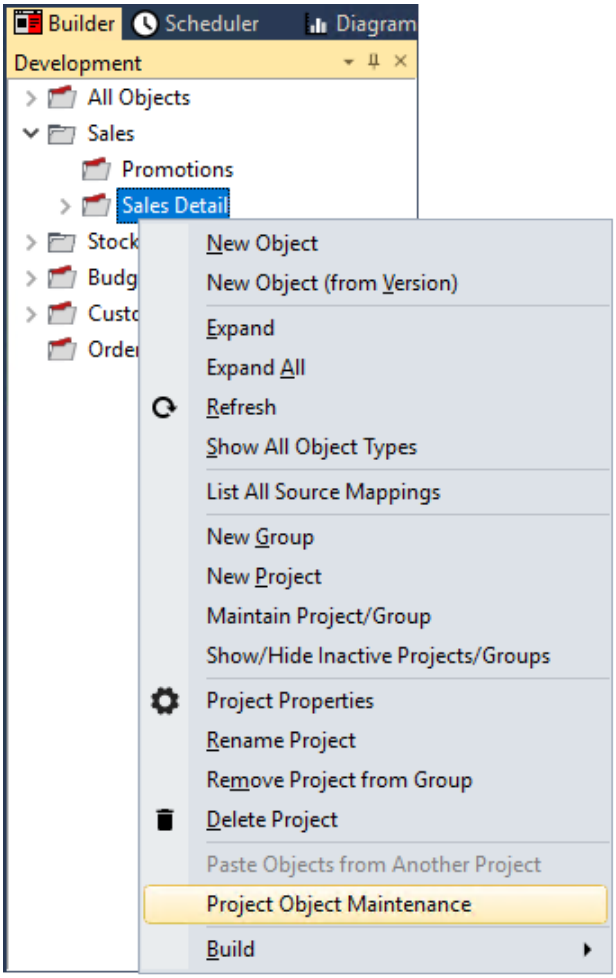
There are several ways to remove objects from a project:

- **Drag** the object to the blank area at the bottom of the pane.
- **Drag** the object into the middle pane.
- **Drag** the object to the **All Objects** project.
- Right-click the object in the left pane and select **Projects > Remove from Project.**
- Highlight several objects in the middle pane, right-click and select **Projects > Remove from Project**.
- **Use the Project Object Maintenance Facility** (Refer to **Using Project Object Maintenance** for details).

Using Project Object Maintenance

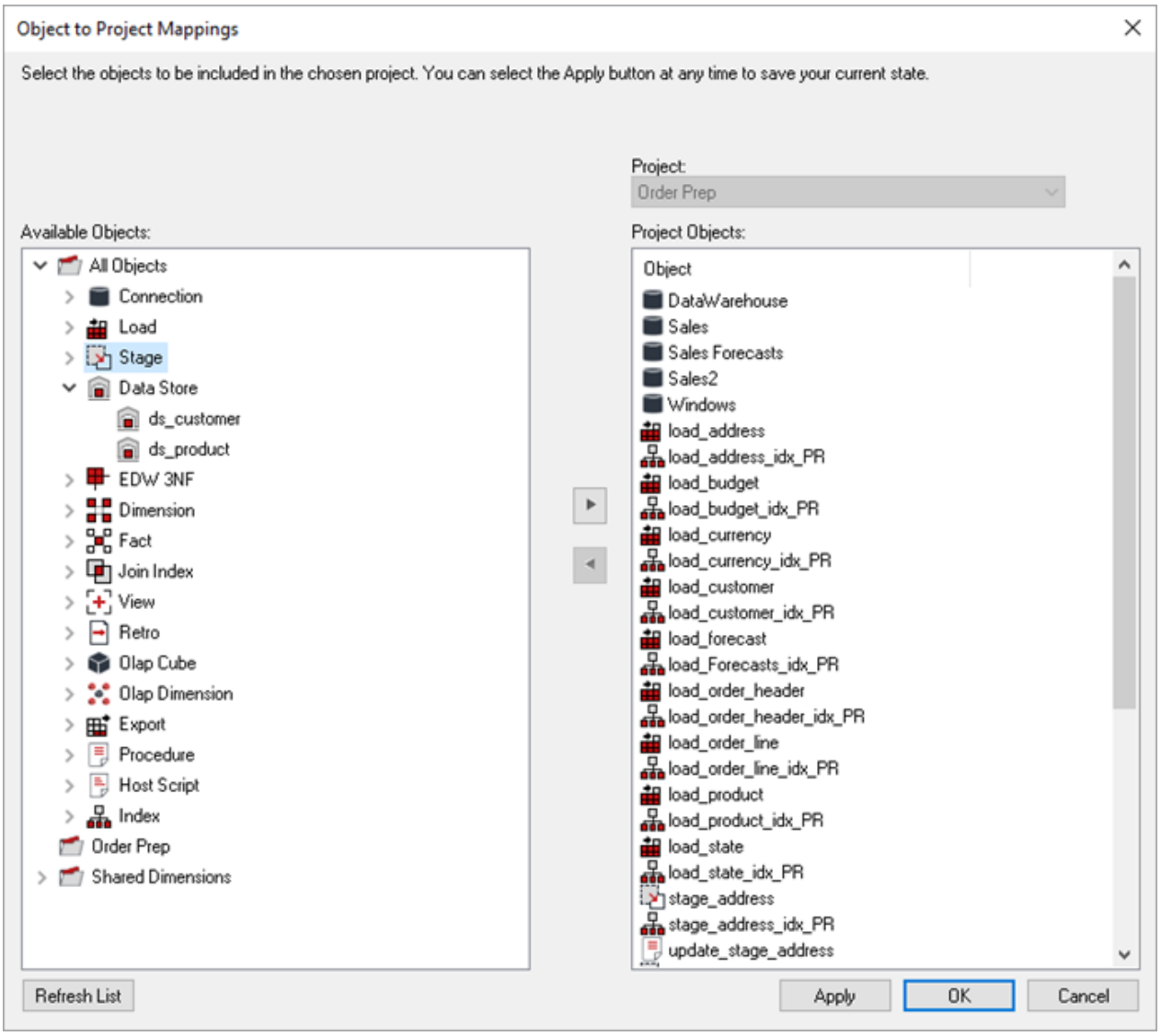
The Project Object Maintenance facility enables you to manage the objects within the different projects defined in your metadata repository.

The **Object to Project Mappings** window is launched by right-clicking a project and selecting **Project Object Maintenance** from the context menu.



© Copyright, WhereScape Software Ltd All rights reserved.

The following window is displayed:



© Copyright, WhereScape Software Ltd All rights reserved.

Adding an Object

1. Select and object in the **Available Objects** pane and click .
   The selected object is moved to the **Project Objects** pane.

   Note:

   By default, associated objects (procedures, scripts and indexes) are also moved to the right. These can be manually removed if not required.
2. Click **OK**.

Removing an Object

1. Select and object in the **Project Objects** pane and click .
   The selected object is moved to the **Available Objects** pane.

   Note:

   By default, associated objects (procedures, scripts and indexes) are also removed.
2. Click **OK**.

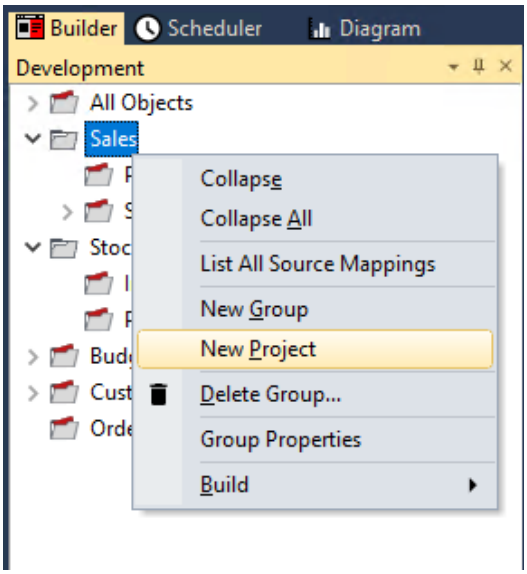
Tips:

- Clicking **Apply** updates object/project relationships without having to exit the Project Object Maintenance facility.
- Clicking **Refresh List** refreshes the object tree in the left pane.

Adding Projects to Groups

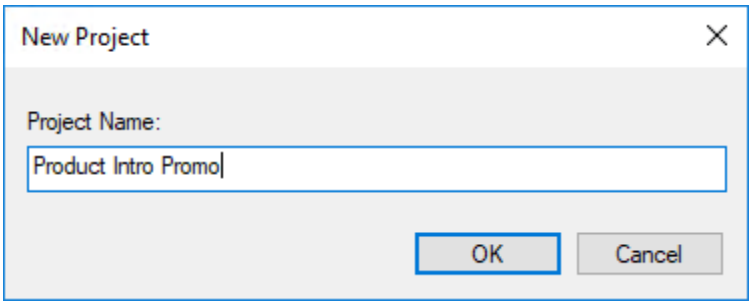
You can add projects to a group by creating the project under the group:

1. Right click the target Group and then select **New Project** from the context menu:

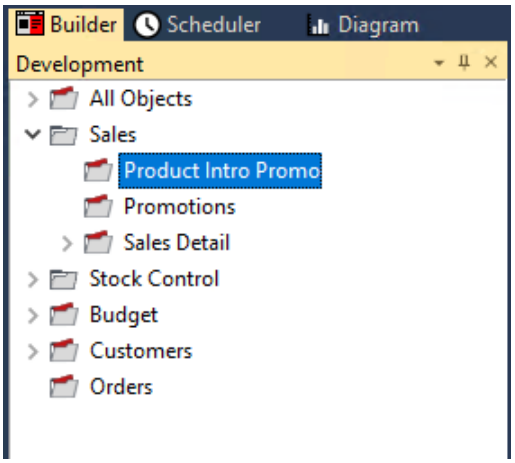


© Copyright, WhereScape Software Ltd All rights reserved.

2. Enter a name in the **New Project** window and then click **OK**.



The new project is added under the selected Group:



Tip:

You can also move an existing project from another group into this group. Refer to **Moving Projects within Groups** for details.

Removing Projects from Groups

A project can be removed from a group by:

- **Dragging** the project to the blank area at the bottom of the pane.
- **Dragging** the project into the middle pane.
- **Dragging** the project to the **All Objects** project.
- **Removing** the project using right-click **Remove Project from Group**.
- **Deleting** the project using right-click **Delete Project**.

Note:

The first four methods move the project out from the group to become an independent project. The last option removes the project from the metadata repository.

© Copyright, WhereScape Software Ltd All rights reserved.

Moving Projects within Groups

Note:

An **object** can be in any number of projects and a **project** can also be included in more than one Project Group.

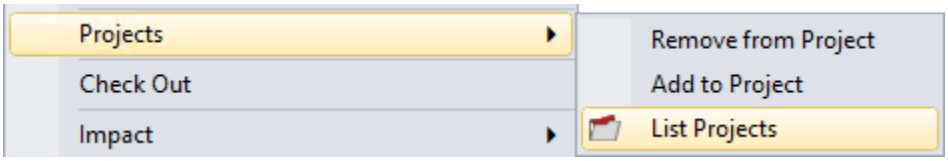
Performing a drag and drop from one group to another group simply creates an additional **Project > Group Mapping**.

To move a project from one group to another group:

1. **Copy** the project by dragging the project from one group to the other group.
2. **Remove** the project from its original group by right-clicking and selecting **Remove Project from Group**.

Listing Project Memberships for Objects

To list projects that contain a specific object, right click the object in the left pane and select **Projects > List Projects**. Results are shown in the bottom pane.



Multiple objects of the same type can be selected by double-clicking the **Object Type** icon in the left pane (e.g. **Connection**, **Load Table**, etc.) and then **Ctrl +** clicking multiple objects displayed in the middle pane.

Tip:

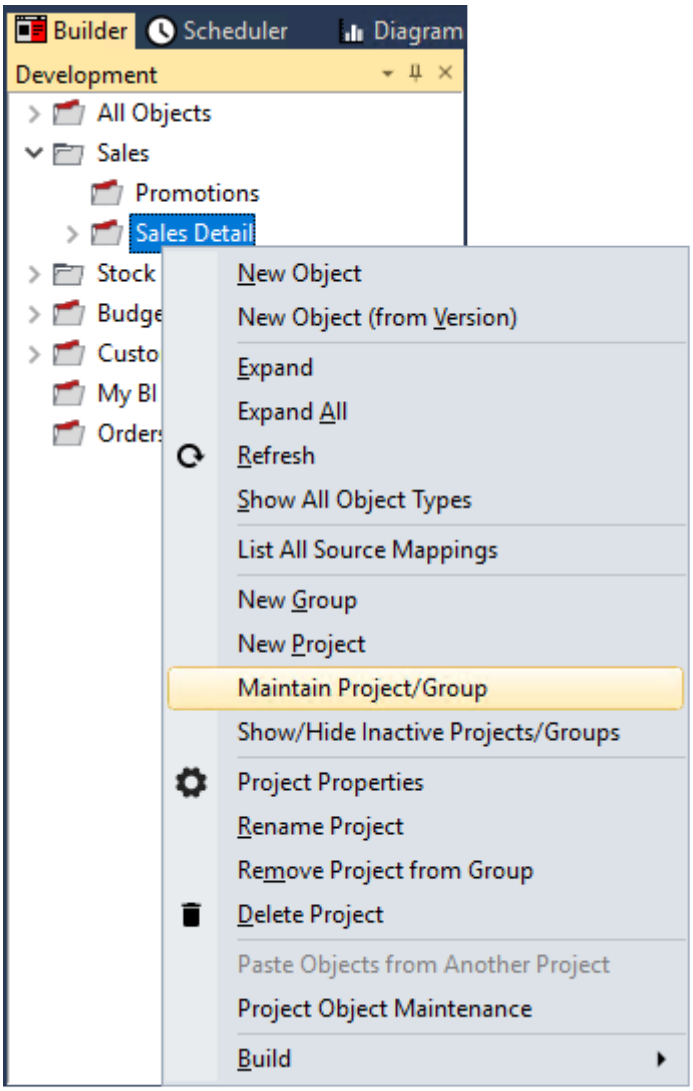
You can double click a Project or Project Group listed in the left pane to see the Project membership details of each object in the selected Project or Project Group in your repository.

© Copyright, WhereScape Software Ltd All rights reserved.

Maintaining Project/Group Active Flag

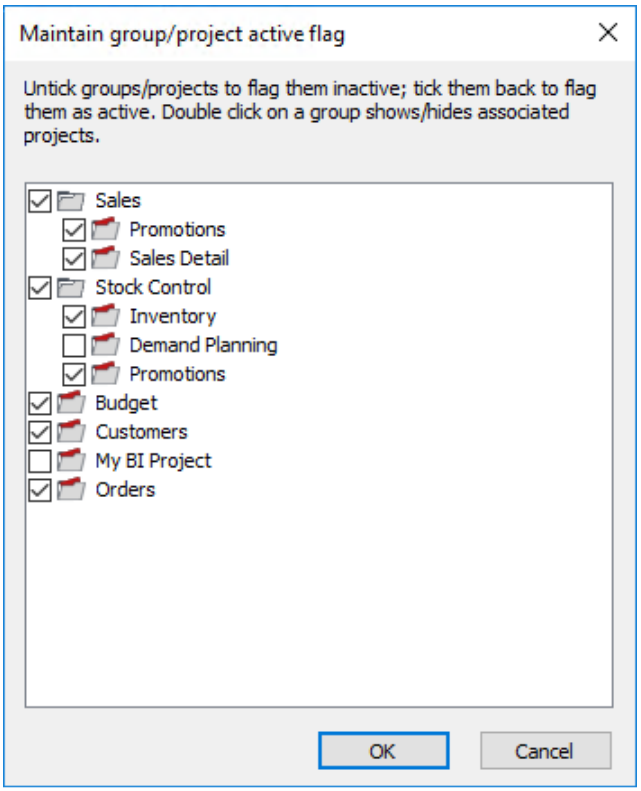
The Maintain Project/Group facility enables you to flag the active status of Projects and Project Groups defined in your metadata repository. This facility enables you to hide Inactive projects that are no longer required, as an alternative to deleting them to preserve information about the contents of old projects.

The **Maintain group/project active flag** window is launched by right-clicking a project from the left pane and selecting **Maintain Project/Group** from the context menu.



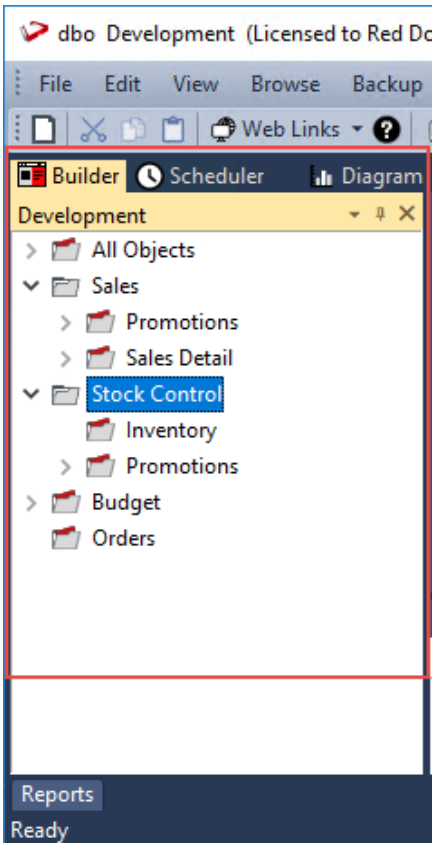
© Copyright, WhereScape Software Ltd All rights reserved.

The following window is displayed:



In the above example, there are two projects that are inactive, the **Demand Planning** project which is grouped under the **Stock Control** project group and the **My BI Project** which is an independent project, e.g. does not belong to a project group.

Because these two projects are flagged as **inactive**, they are not displayed in the left pane of the **Builder** window.



© Copyright, WhereScape Software Ltd All rights reserved.

Notes:

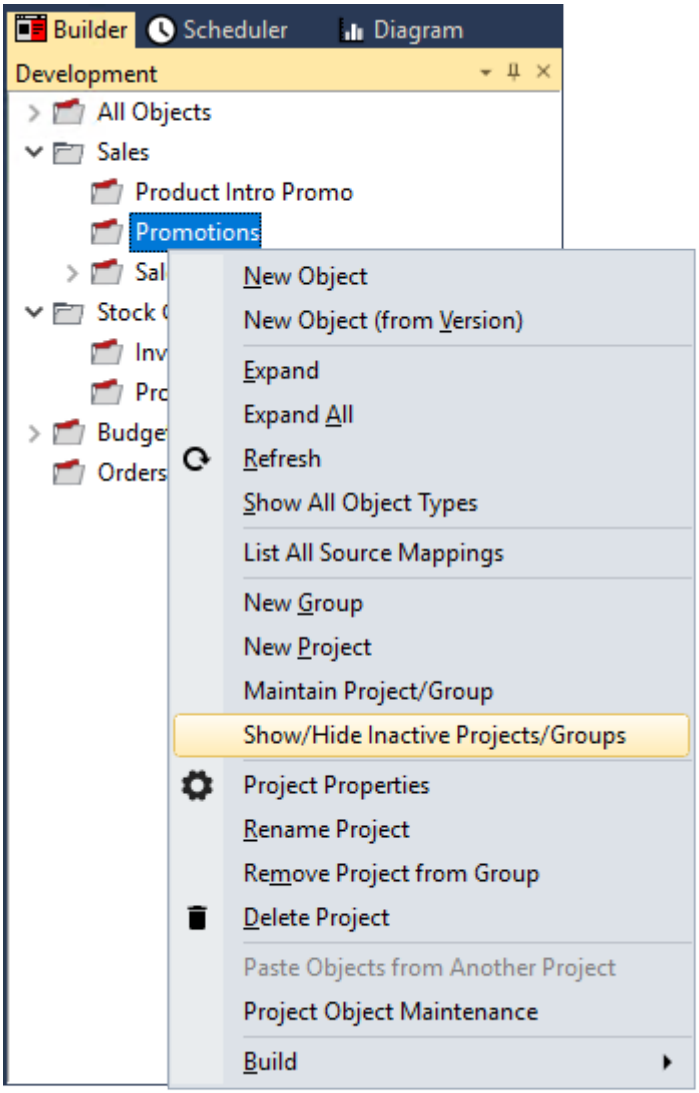
If a Project that belongs to several Groups is flagged as inactive, it becomes inactive on all the Groups it exists.
Projects/Groups that are flagged as inactive are also not displayed/available in the following:

- Generated documentation and diagrams, refer to **Documentation and Diagrams** for more information.
- **Build Deployment Application > Objects to Add/Replace**, refer to **Application Creation** for more information.

Displaying/Hiding Inactive Projects/Groups

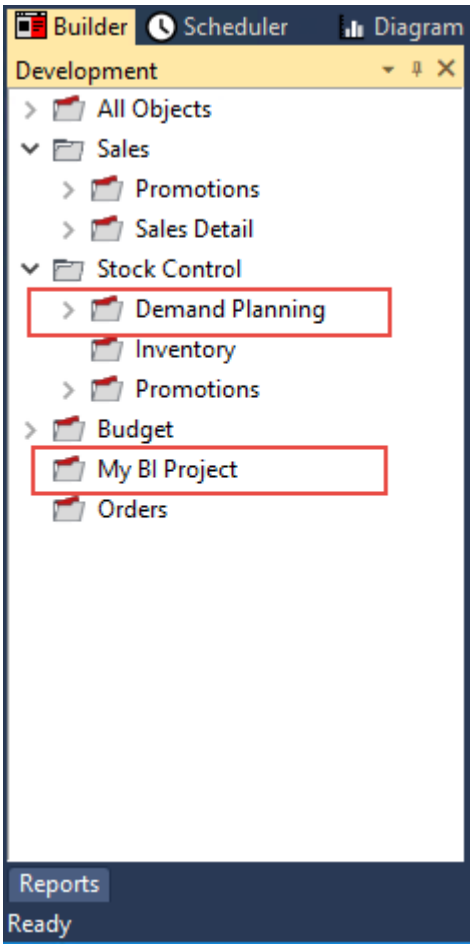
The **Show/Hide Inactive Projects/Groups** function enables you to control the display of the projects or project groups that have been flagged as inactive, using the **Maintain group/object active flag** window.

To display the inactive projects/groups in the left pane of the **Builder** window, right-click a project and then select **Show/Hide Inactive Projects/Groups** from the context menu:



© Copyright, WhereScape Software Ltd All rights reserved.

The projects flagged inactive and previously hidden are displayed in the left pane of the **Builder** window:



In the same manner, use **Show/Hide Inactive Projects/Groups** to hide the inactive projects/groups that are displayed in the left pane of the **Builder** window.

© Copyright, WhereScape Software Ltd All rights reserved.

Building Scheduler Jobs

Scheduler jobs can be created from Object Groups, Projects and Project Groups defined in RED via the **Build > Build Job** context menu. The detailed steps are described in the following sections.

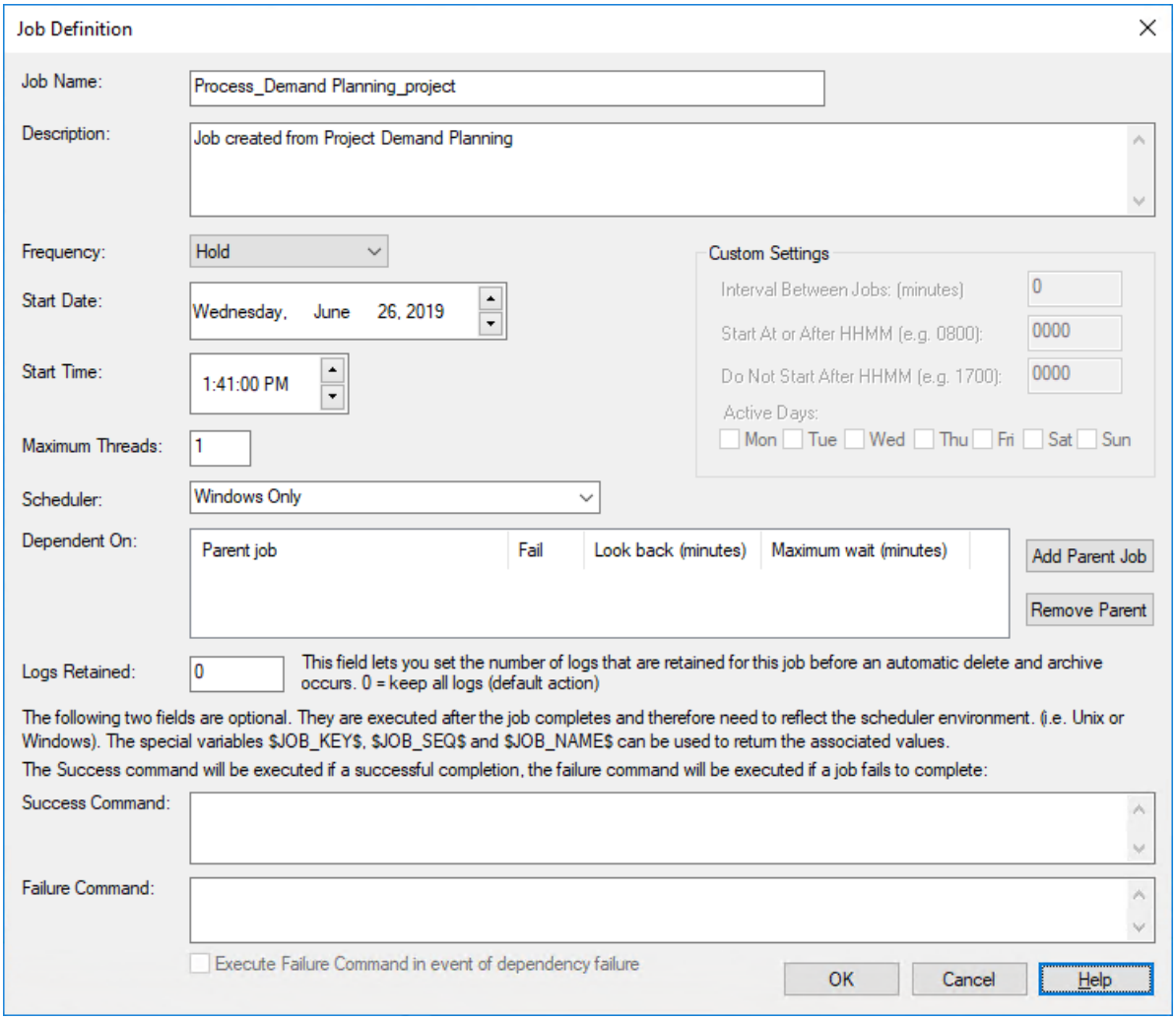
- Building Scheduler Jobs from Projects or Groups
- Building Scheduler Jobs from Object Groups

Building Scheduler Jobs from Projects or Groups

WhereScape RED enables you to automatically create a Scheduler job from the table objects in a Project or Project Group defined in RED. This feature utilizes the track-back information for all the objects included in a Project or Project Group to determine their task dependencies and then creates the Scheduler job.

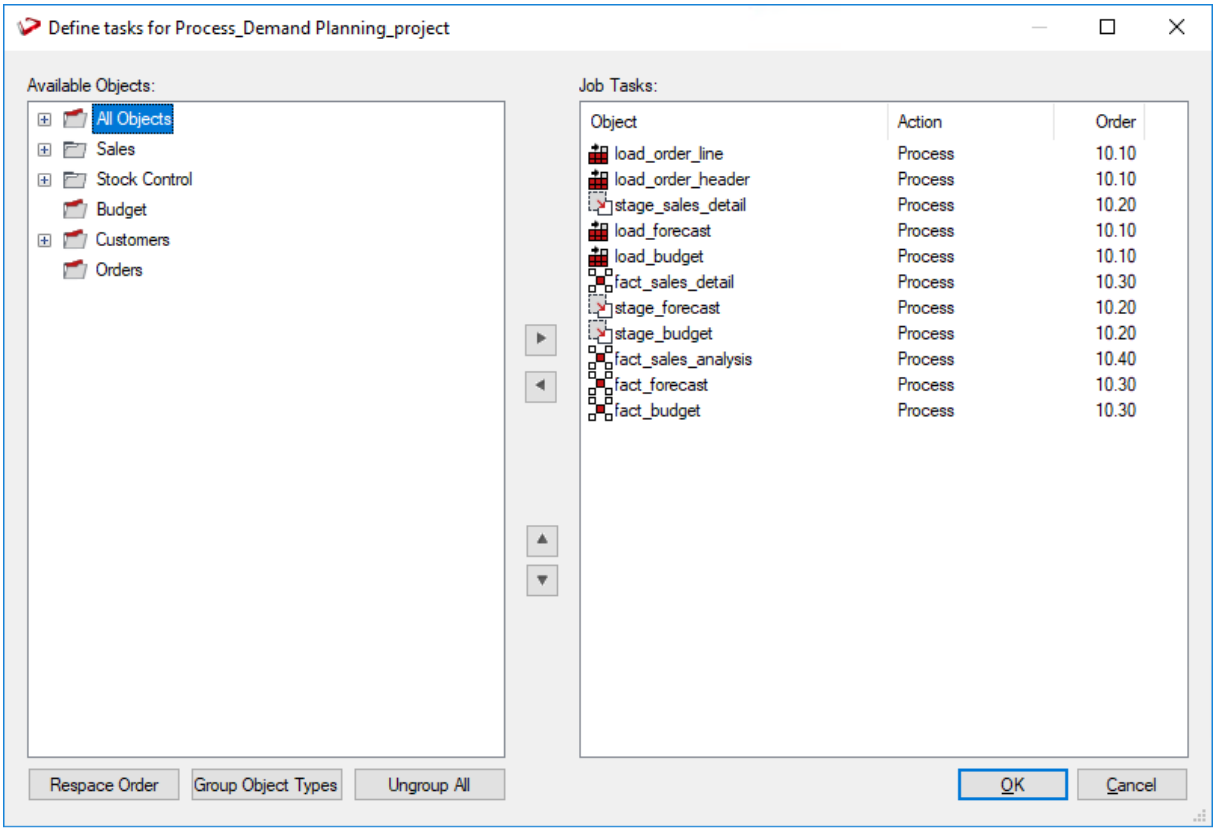
The steps are outlined below:

1. Right click the **Project** or **Group** from the left pane of the **Builder** window.
2. Select the **Build > Build Job** option from the context menu.
3. Make any changes to the details and settings in the **Job Definition** window as required and then click **OK** to define the tasks in the Scheduler job.

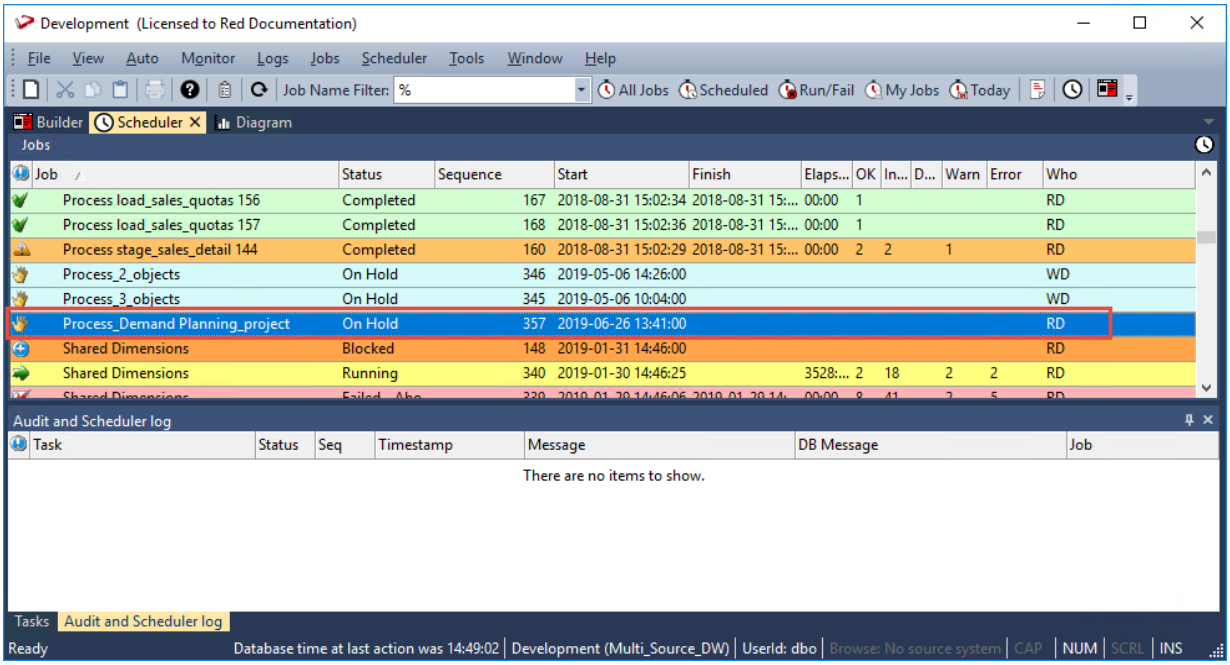


© Copyright, WhereScape Software Ltd All rights reserved.

4. Review the **Job Tasks** that have been defined for the job and remove tasks you want to exclude as required.



5. Click **OK** to create the job, the **Scheduler** window is displayed.
Refresh the screen to see the created job, its status and other details.



Refer to **Creating a Job** for details about the fields and settings in the **Job Definition** and **Define tasks** windows.

Note:

The **Build** context menu is not available for the **All Objects** project, listed in the left pane of the **Builder** window.

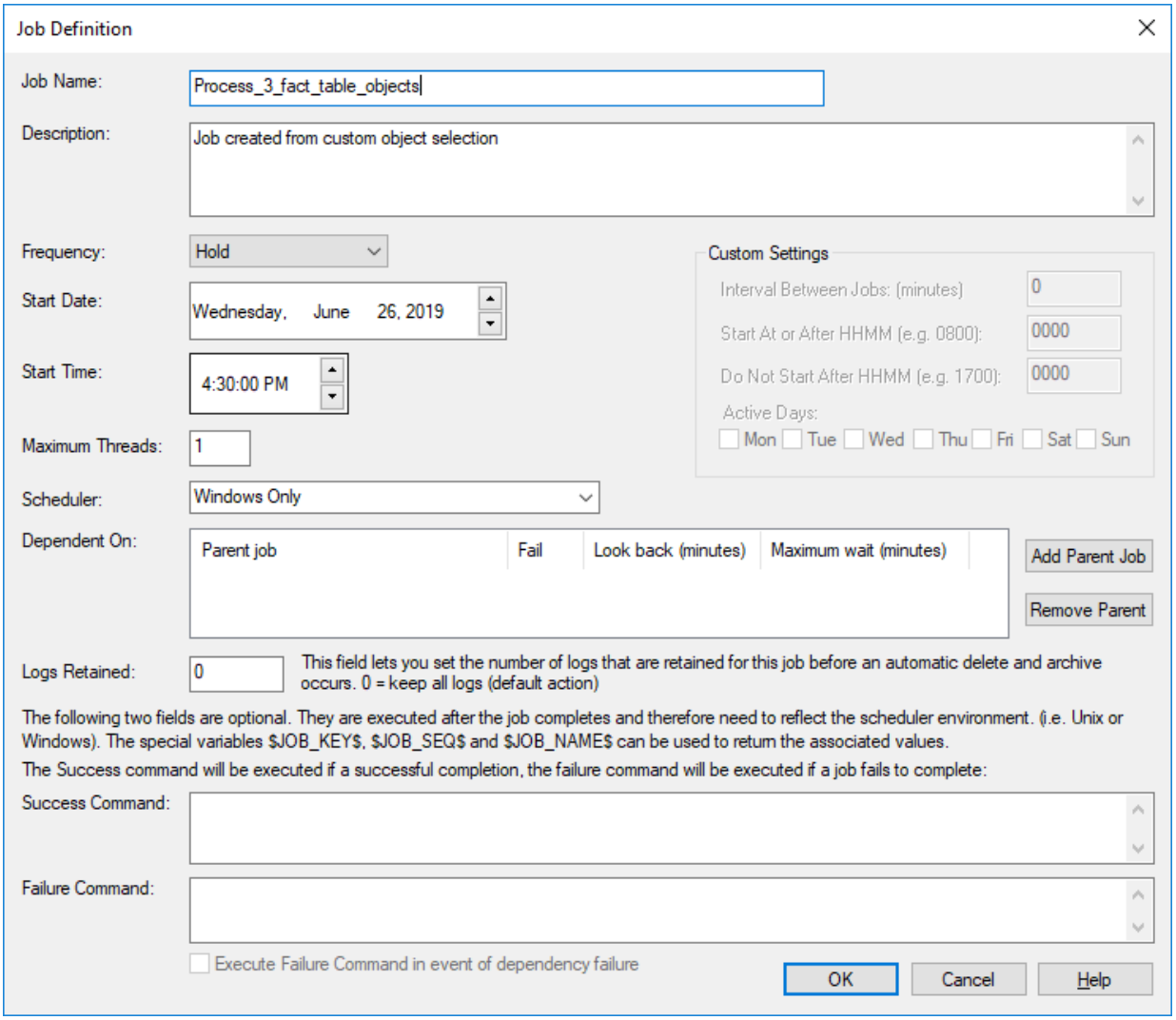
© Copyright, WhereScape Software Ltd All rights reserved.

Building Scheduler Jobs from Object Groups

WhereScape RED enables you to automatically create a **Scheduler job** from table objects in an Object Group defined in RED. This feature utilizes the track-back information for the objects selected in an Object Group to determine their task dependencies and then creates the Scheduler job.

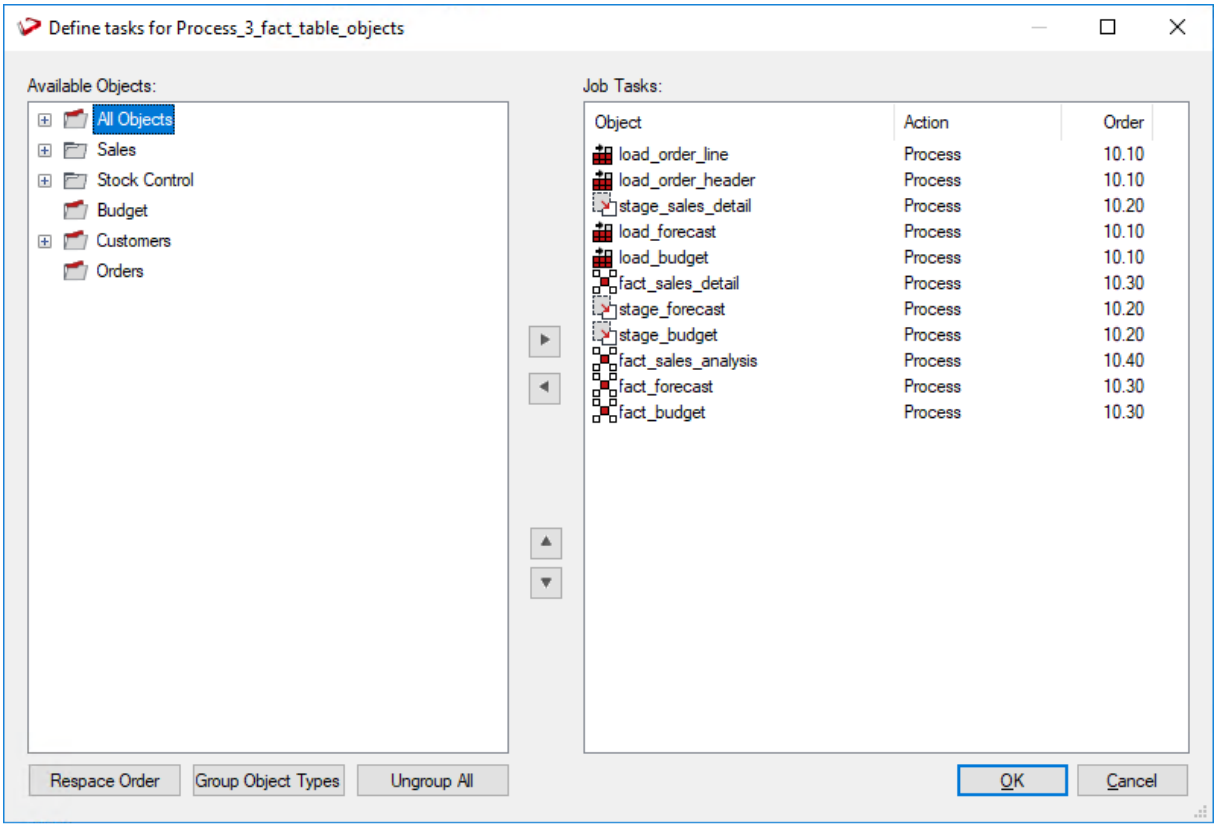
The steps are outlined below:

1. Double click the Object Group name (e.g. Fact Table, Stage Table, etc.) from the left pane of the **Builder** window to display all the tables in the selected Object Group in the middle pane.
2. Select a table or multiple tables in the middle pane that you want to include in the Scheduler Job.
3. Right-click a highlighted table and then select the **Build > Build Job** option from the context menu.
4. Type in the **Job Name** and make any changes to the details and settings in the **Job Definition** window as required and then click **OK** to define the tasks in the Scheduler job.

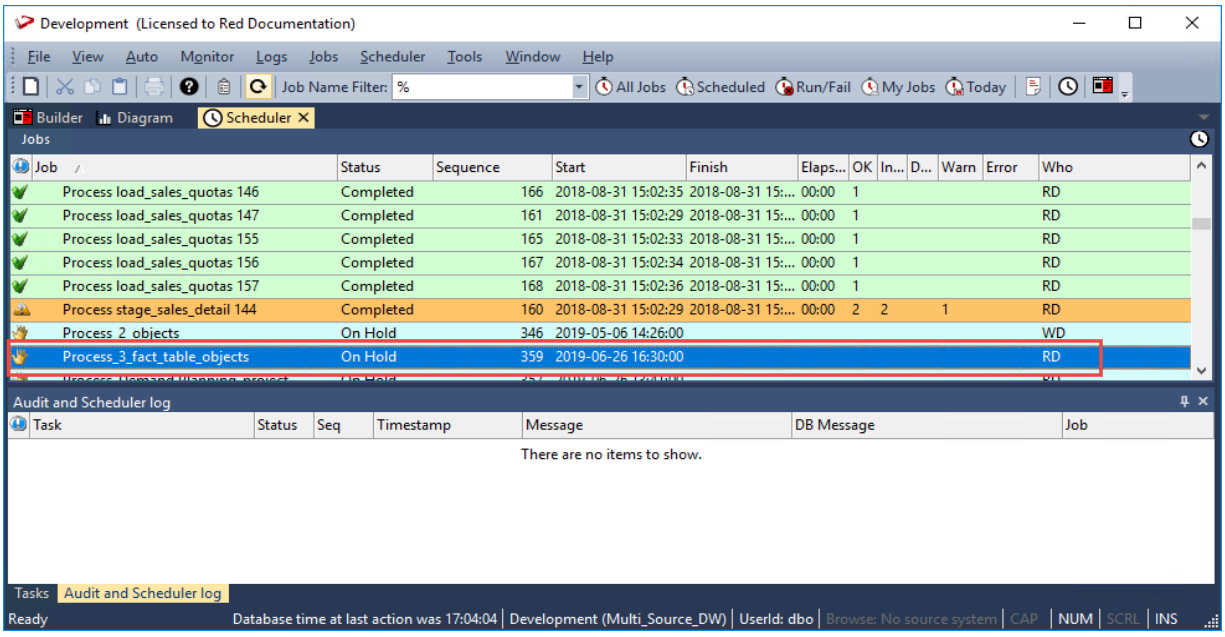


© Copyright, WhereScape Software Ltd All rights reserved.

5. Review the **Job Tasks** that have been defined for the job and remove tasks you want to exclude as required.



6. Click **OK** to create the job, the **Scheduler** window is displayed.
Refresh the screen to see the created job, its status and other details.



Refer to **Creating a Job** for details about the fields and settings in the **Job Definition** and **Define tasks** windows.

© Copyright, WhereScape Software Ltd All rights reserved.

Building Applications

Deployment applications can be created from Object Groups, Projects and Project Groups defined in RED via the **Build > Build Job** context menu. The detailed steps are described in the following sections.

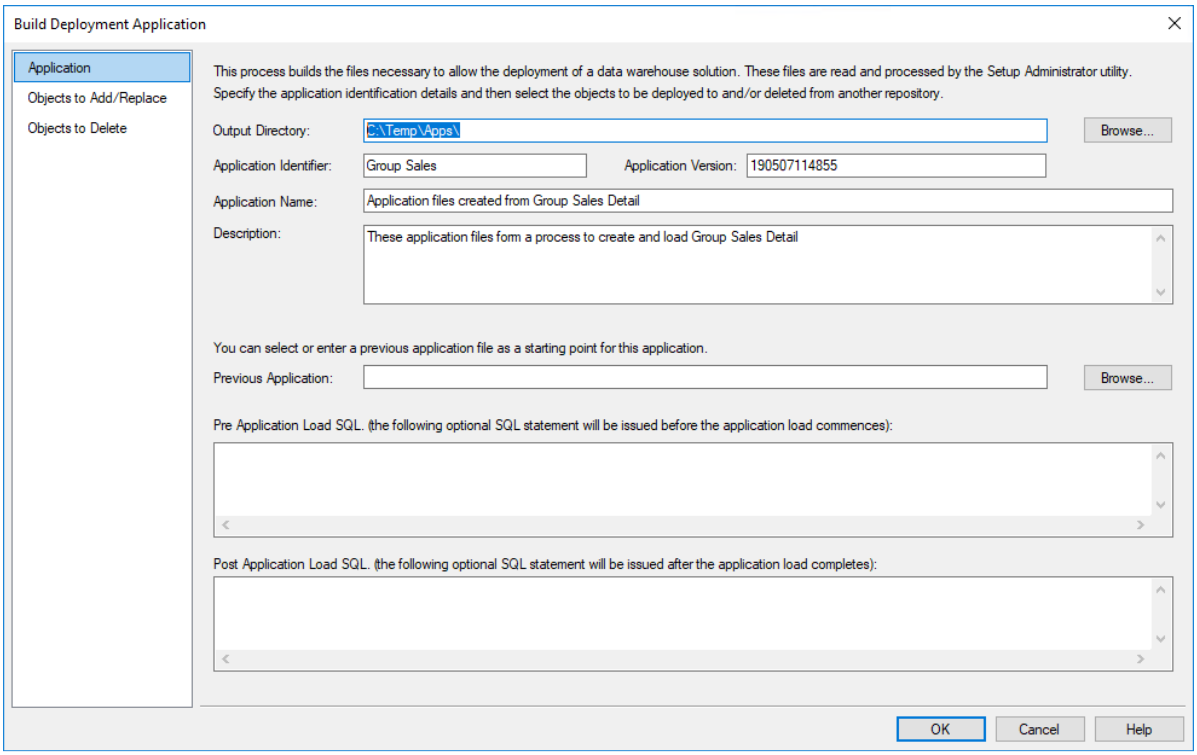
- Building Applications from Projects or Groups
- Building Applications from Object Groups

Building Applications from Projects or Groups

WhereScape RED enables you to automatically create a Deployment Application from all the table objects in a Project or Project Group defined in RED. This feature utilizes the track-back information for all the objects included in a Project or Project Group to determine the objects to be deployed and their task dependencies; and then creates the Application.

The steps are outlined below:

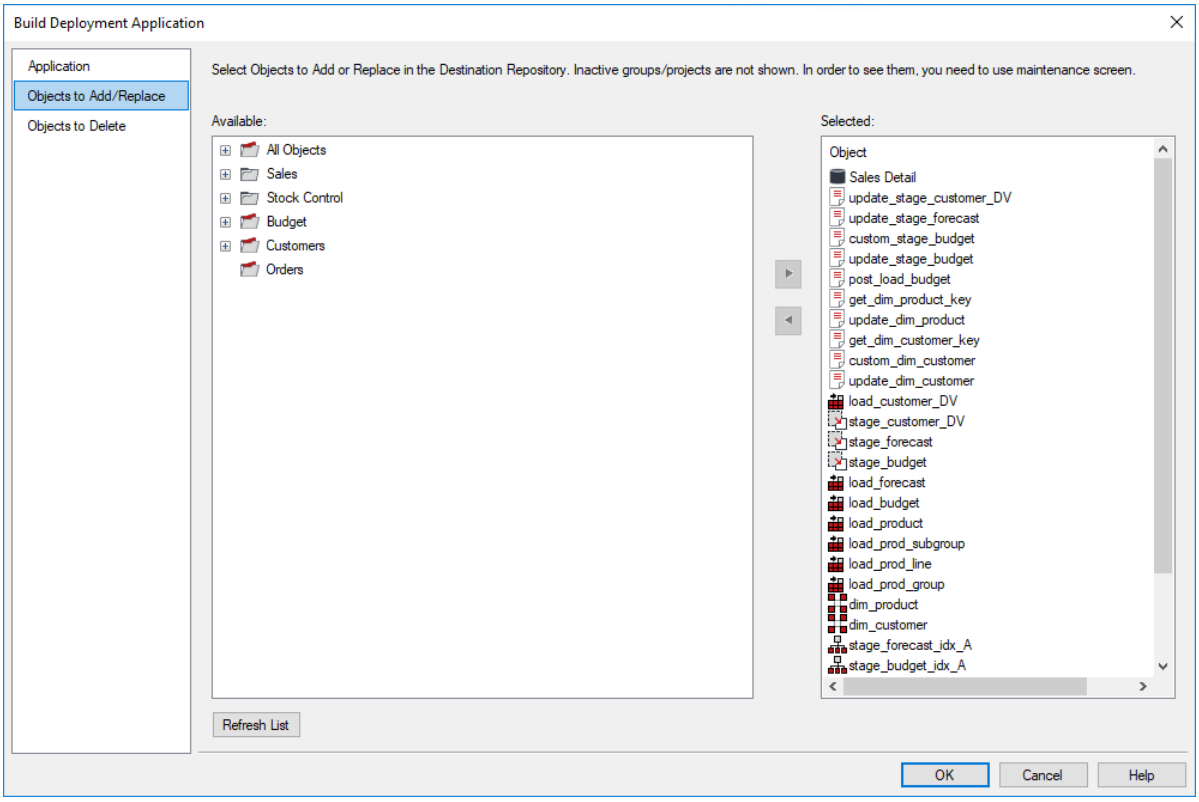
1. Right click the **Project** or **Group** from the left pane of the **Builder** window.
2. Select the **Build > Build Application** option from the context menu.
3. Make any changes to the details and settings in the **Build Deployment Application** window as required.



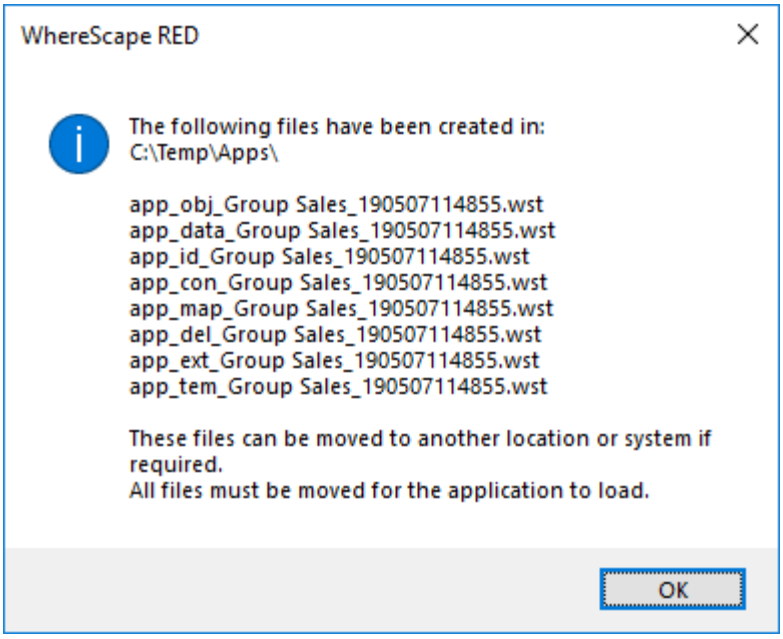
Refer to Application Creation for details about the fields and settings in the **Build Deployment Application** window.

© Copyright, WhereScape Software Ltd All rights reserved.

4. Click the other tabs to check or change the selected objects to be deployed in the target metadata repository



5. Click **OK** to create the application. A message is displayed to confirm the creation of the application files.



Note:

The **Build** context menu is not available for the **All Objects** project, listed in the left pane of the **Builder** window.

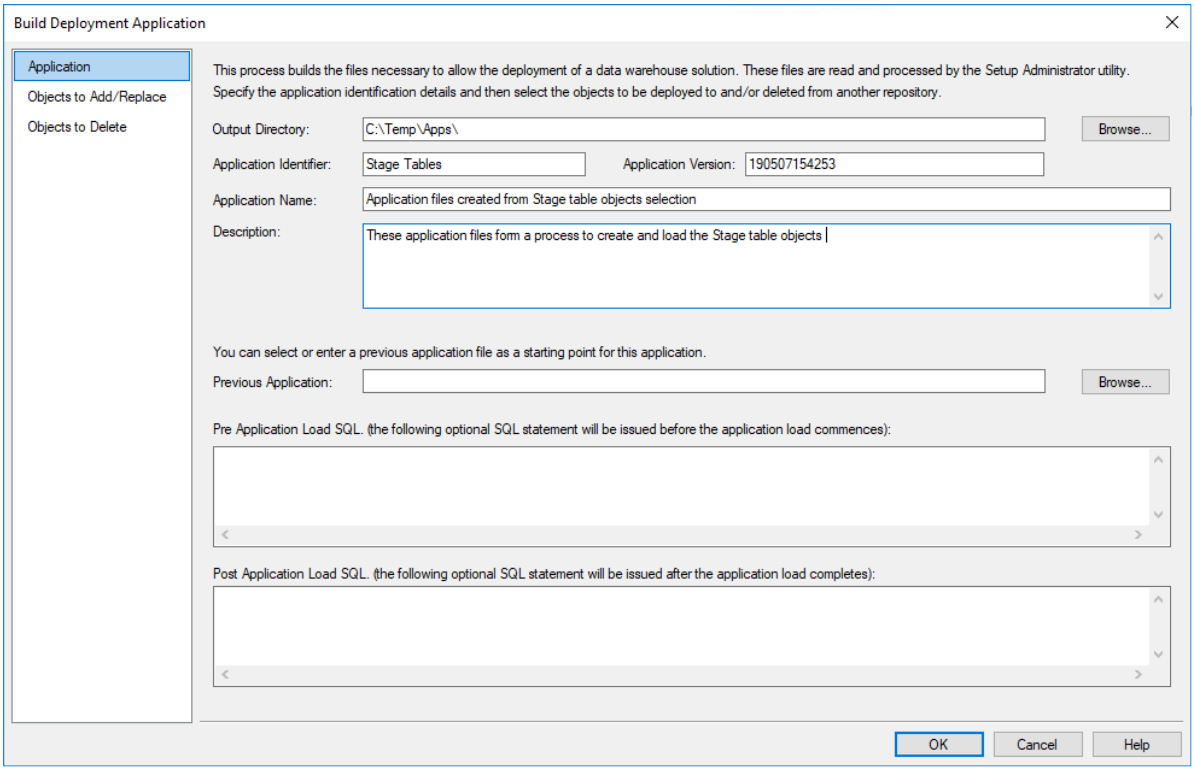
© Copyright, WhereScape Software Ltd All rights reserved.

Building Applications from Object Groups

WhereScape RED enables you to automatically create a **Deployment Application** from table objects in an Object Group defined in RED. This feature utilizes the track-back information for all the objects included in an Object Group to determine the objects to be deployed and their task dependencies; and then creates the Application.

The steps are outlined below:

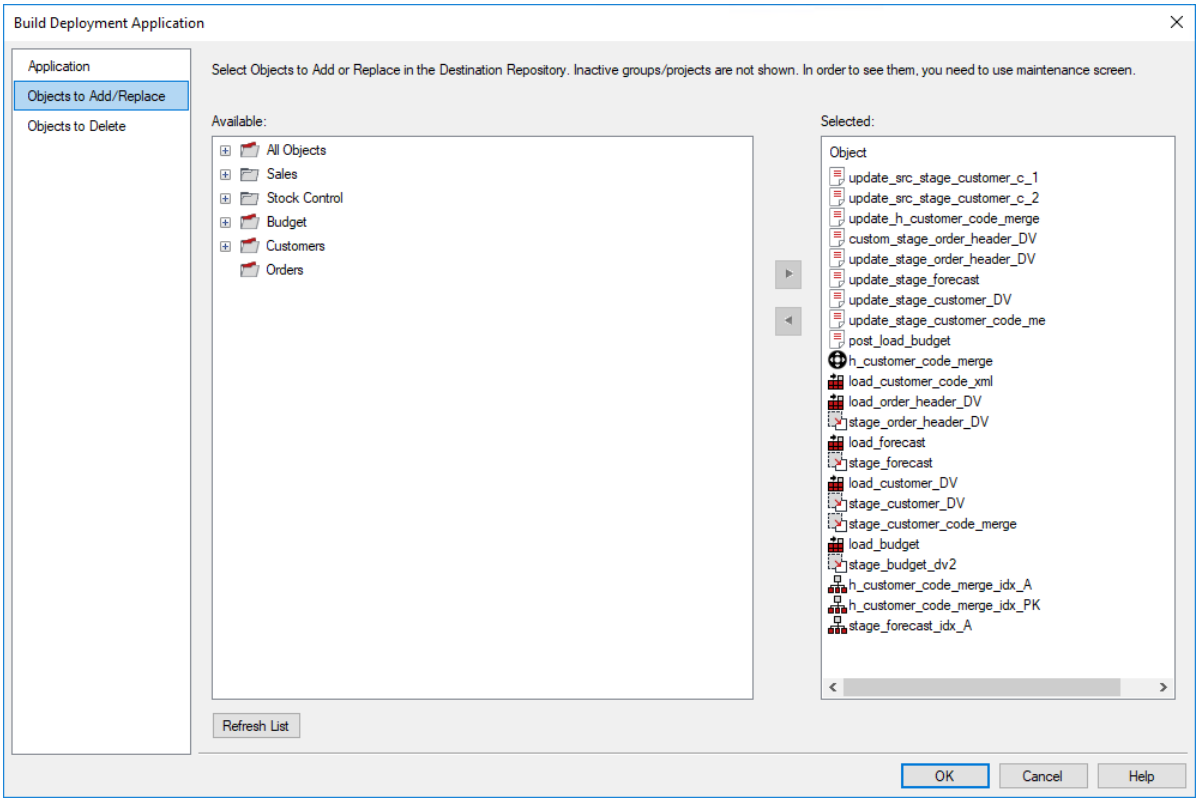
1. Double click the Object Group name (e.g. Load Table, Stage Table, etc.) from the left pane of the **Builder** window to display all the tables in the selected Object Group in the middle pane.
2. Select a table or multiple tables in the middle pane that you want to include in the Application.
3. Right-click a highlighted table and then select the **Build > Build Application** option from the context menu.
4. Type in the **Application Identifier** and make any changes to the details and settings in the **Build Deployment Application** window as required.



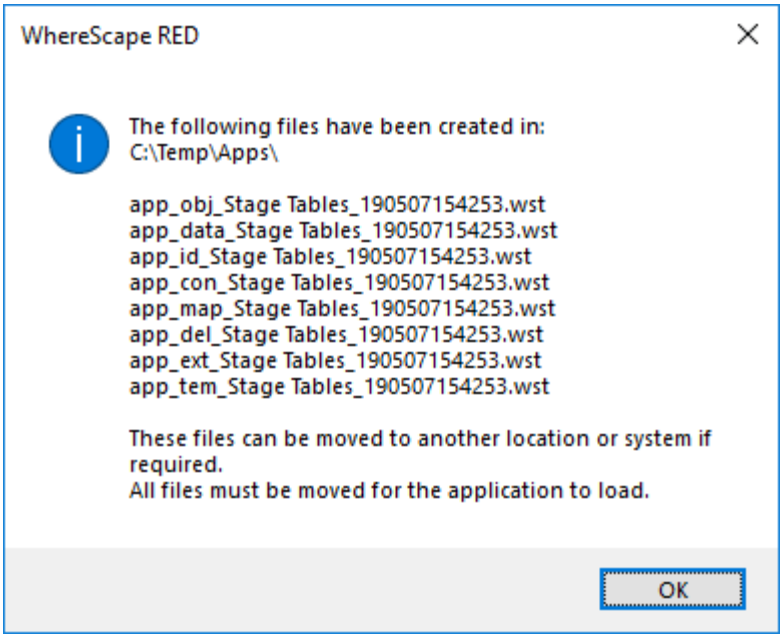
Refer to **Application Creation** for details about the fields and settings in the **Build Deployment Application** window.

© Copyright, WhereScape Software Ltd All rights reserved.

5. Click the other tabs to check or change the selected objects to be deployed in the target metadata repository



6. Click **OK** to create the application. A message is displayed to confirm the creation of the application files.



© Copyright, WhereScape Software Ltd All rights reserved.

Windows and Panes

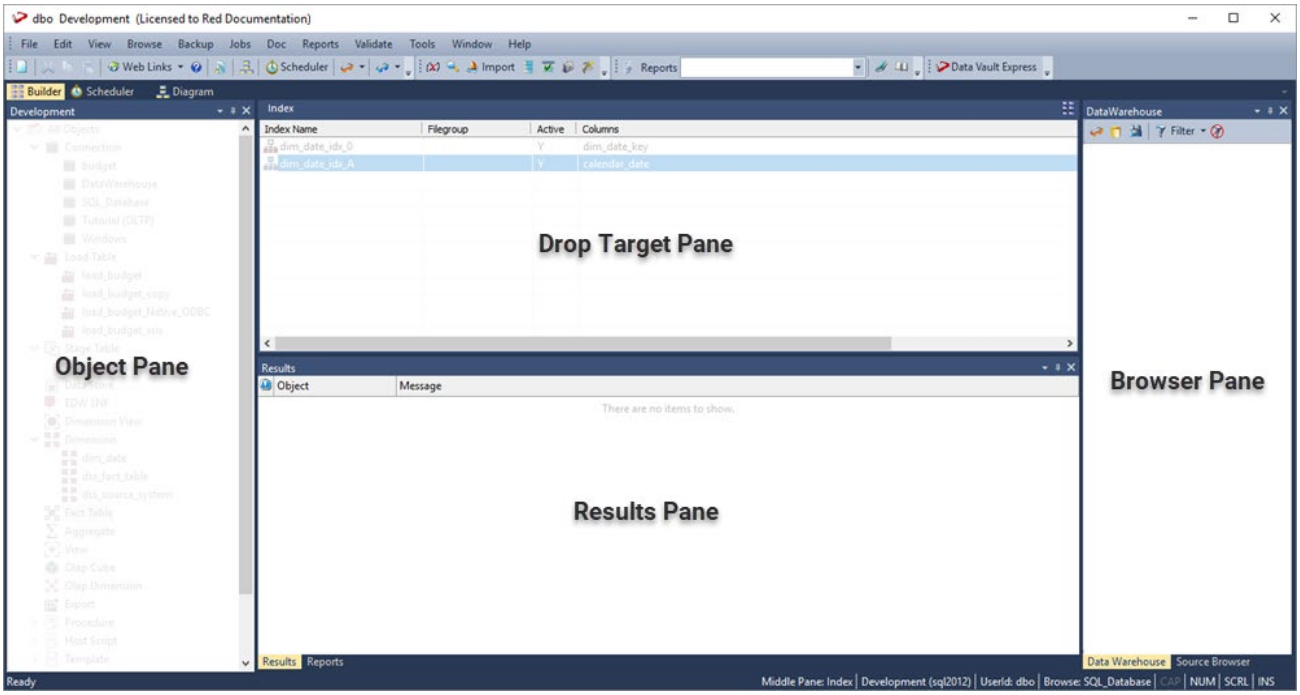
WhereScape RED has a number of different windows that are utilized in the building and maintenance of a data warehouse. Each window, in some cases are divided into panes.

There are four main windows in RED that are used extensively in building a data warehouse:

- The **Builder Window**.
- The **Scheduler Window**.
- The **Diagram Window**.
- The **Procedure Editor Window**.

Builder Window

The Builder window has four panes.

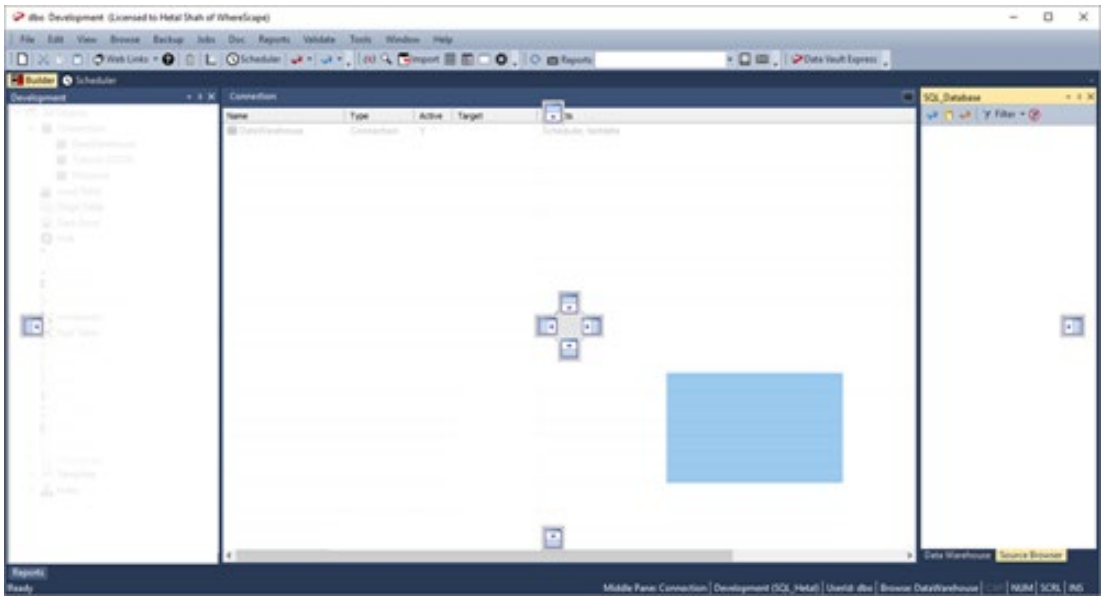


Panes	Description
Object Pane	This pane contains all the objects in the metadata repository. These objects are stored in object groups (e.g. Dimension, Fact, etc.). The object groups in turn can optionally be stored in Projects, and the Projects can optionally be stored within Project Groups.
Drop Target Pane	This pane is used to display the results of various queries on both the metadata and the underlying source and database tables. The middle pane is also used as the drop target in drag and drop operations. The status line at the bottom of the screen displays the current contents of the middle pane. To send the middle pane output to a file or to clipboard, refer to **Export Middle Pane Output** for details.
Results Pane	This pane is used to display the results of various queries on both the metadata and the underlying source and database tables.
Browser Pane	This pane displays both Source and Data Warehouse systems. There are two browser panes available at any one time. The source may be the data warehouse itself. The Browser Pane can be filtered: • To filter by type, click the down arrow next to the **Filter** button. • To filter by name, click the **Filter** button and enter the filter criteria in the **Filter** dialog. Name based filters can be cleared by clicking the down arrow next to the **Filter** button and selecting **Clear Name Based Filter** from the menu.

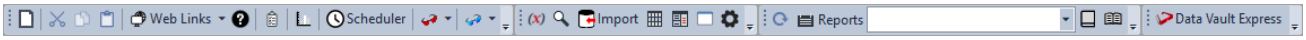
© Copyright, WhereScape Software Ltd All rights reserved.

Tips:

- Pop-up menus are available in all four panes. Use **Ctrl+M** to display the pop-up menu when an item is selected in any of the panes.
- You can select multiple items displayed in the drop target/middle pane.
- **F5** or **Ctrl+R** can be used to refresh the left and right panes, when the cursor is positioned within these panes.
- The left, bottom and right panes can be dragged out and docked elsewhere. Docking handles appear when a pane is dragged which you can use to specify the location for the pane, as shown below:



Toolbar



Icon	Description
	Create a New Object (Ctrl+N) - Click to create a new object.
	Web Links - Click to display the online WhereScape forum in a new tab. To select or enter other web links, click the down arrow beside the Web Links button and select an URL from the list. **Tip:** Select **Configure** to open the following dialog box, which enables you to define up to 3 custom URLs. Enter the name and the URL for the custom link. Click **OK**.
	View application documentation - Click to display WhereScape RED help file.
	Open the ToDo List - Click to open a free form text to do list.
	Open the Diagram View - Click to switch to the Diagram view window which provides a graphical representation of the star schema and track back diagrams.
	Open the Scheduler View - Click to switch to the Scheduler control window. Refer to **Scheduler Window** for details.

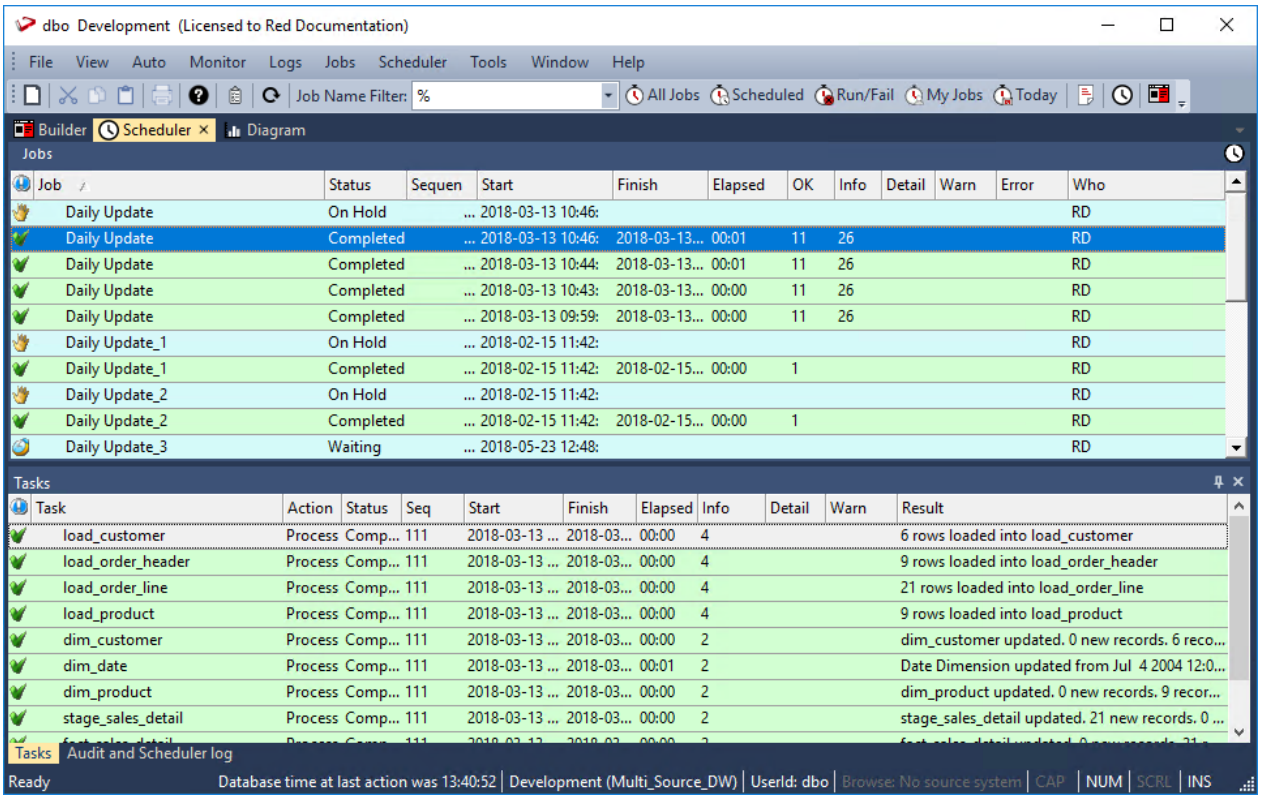
© Copyright, WhereScape Software Ltd All rights reserved.

Icon	Description
	Browse source tables/ files in the right pane (normally Data Warehouse) - Click to invoke the source browser which populates the Browser pane. **Tip:** The source browser, when selected, browses to the connection last used for that button. To change the connection being browsed, click the down arrow beside the glasses icon. Refer to **Browsing a Connection** for details.
	Browse tables/ files in the right pane (normally Source system) - Click to invoke the source browser which populates the Browser pane. **Tip:** The source browser, when selected, browses to the connection last used for that button. To change the connection being browsed, click the down arrow beside the glasses icon. Refer to **Browsing a Connection** for details.
	Display parameters Click to display all parameters (variables).
	Search - Click to search for a string in the metadata.
	Import - Click to import objects from another repository.
	Create a version of all objects - Click to create a version (copy) of all objects.
	Validate tables - Click to compare metadata to database tables.
	Build Deployment Application - Click to build the objects required to load a Deployment Application via the RED Setup Administrator.
	Options - Click to change repository settings.
	Refresh the last ran report and update output in the Reports pane
	Run report against the data in the Data Warehouse
	Create documentation based on the Data Warehouse - Click to create HTML documentation.
	Display the current documentation for the Data Warehouse
	Open the Data Vault Express web page within this application

© Copyright, WhereScape Software Ltd All rights reserved.

Scheduler Window

The **Scheduler Window** is used as the main interface to the scheduler. Jobs can be scheduled, monitored, edited and deleted through this window.



Toolbar

Icon	Description
	New Job - Click to create a new scheduled job.
	Application Help - Click to display WhereScape RED help file.
	Open the ToDo List - Click to open a free form text to do list.
	Auto Refresh of Display - Click to refresh the current (right pane) display every 10 seconds. Click again to stop the auto refresh. The refresh interval can be adjusted through the menu option **Auto/Refresh** interval.
	Job Name Filter - Optional Job name criteria to be applied via the core job selection criteria to further filter jobs.
	All Jobs - Click to view all jobs.
	Scheduled - Click to view scheduled jobs.
	Run/ Fail - Click to view jobs that run/failed.
	My Jobs - Click to view jobs you created.
	Today - Click to view jobs scheduled for today.
	Recent Log Entries - Click to view recent log entries.
	Scheduler Status - Click to view scheduler status.

© Copyright, WhereScape Software Ltd All rights reserved.

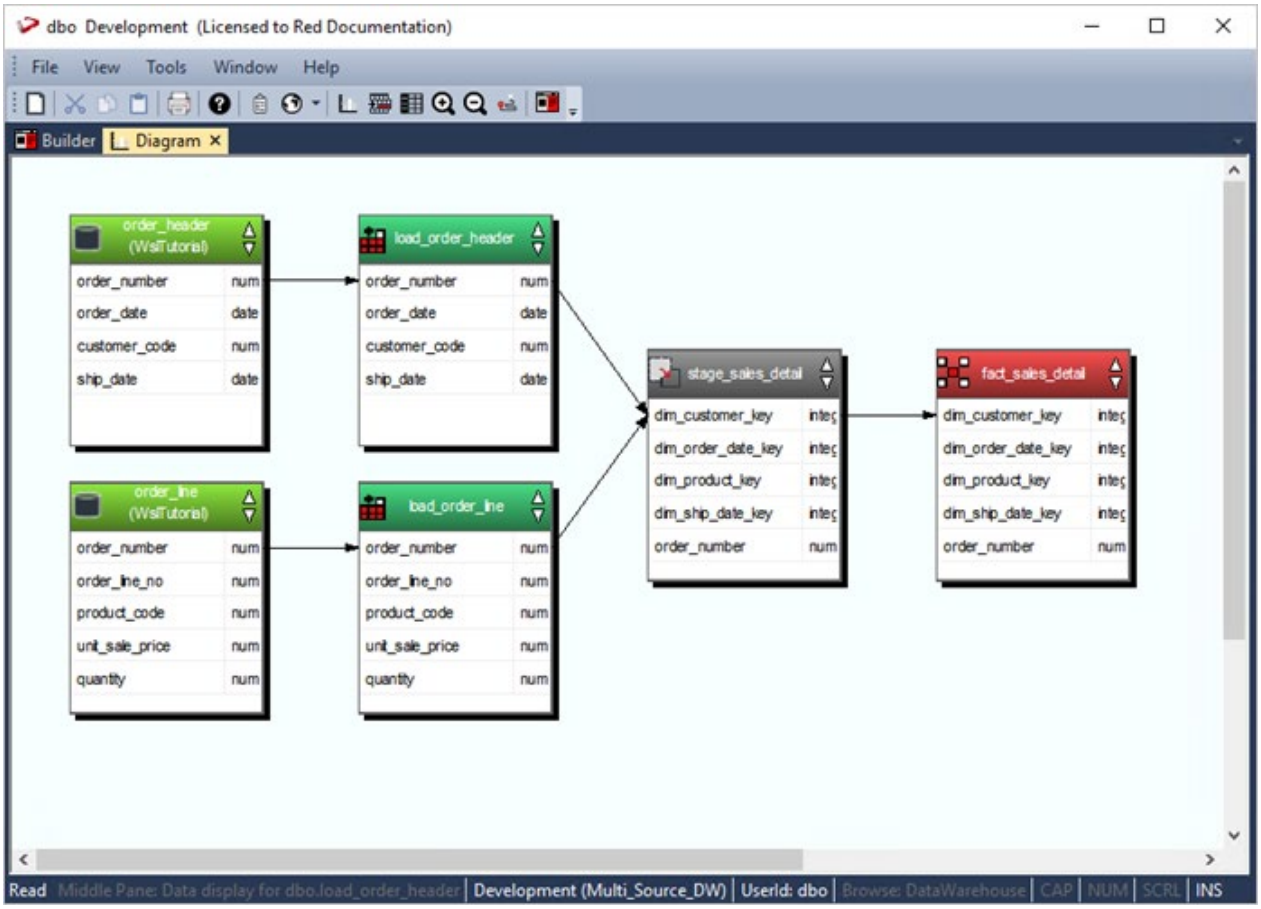
Icon	Description
	Builder Window - Click to switch to the **Builder Window**.

- Double-click the job in the top pane to see the tasks of the selected job in the bottom pane.
- Double-click the task in the bottom pane to see the audit trail displayed in a separate tab in the bottom pane.

Refer to **Scheduler** for more details.

Diagram Window

The **Diagram window** is used to display the tables of the data warehouse in diagrammatic form, showing the various sources or targets of the selected object.



The Diagram Selection is as follows:

- Schema Diagram
- Source Diagram
- Joins Diagram
- Links Diagram
- Impact Diagram
- Dependency Diagram

Refer to **Diagrams** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

Toolbar

Icon	Description
	New Diagram - Click to select a diagram type and table.
	Help - Click to display WhereScape RED help file.
	Open the ToDo list- Click to open a free form text to do list.
	Display overview diagram - Click to display a diagram showing the various objects in the WhereScape metadata and the standard flow of data through these objects. Repeated clicking of the **Overview** button steps through each stage of the data flow.
	Toggle View - Click to toggle between standard and detail view.
	Toggle grid lines - Click to toggle printing of grid lines.
	Zoom in - Click to Zoom In.
	Zoom out - Click to Zoom Out.
	Reroute joins - Click to reroute the connections.
	Builder Window - Click to switch to the **Builder Window**.

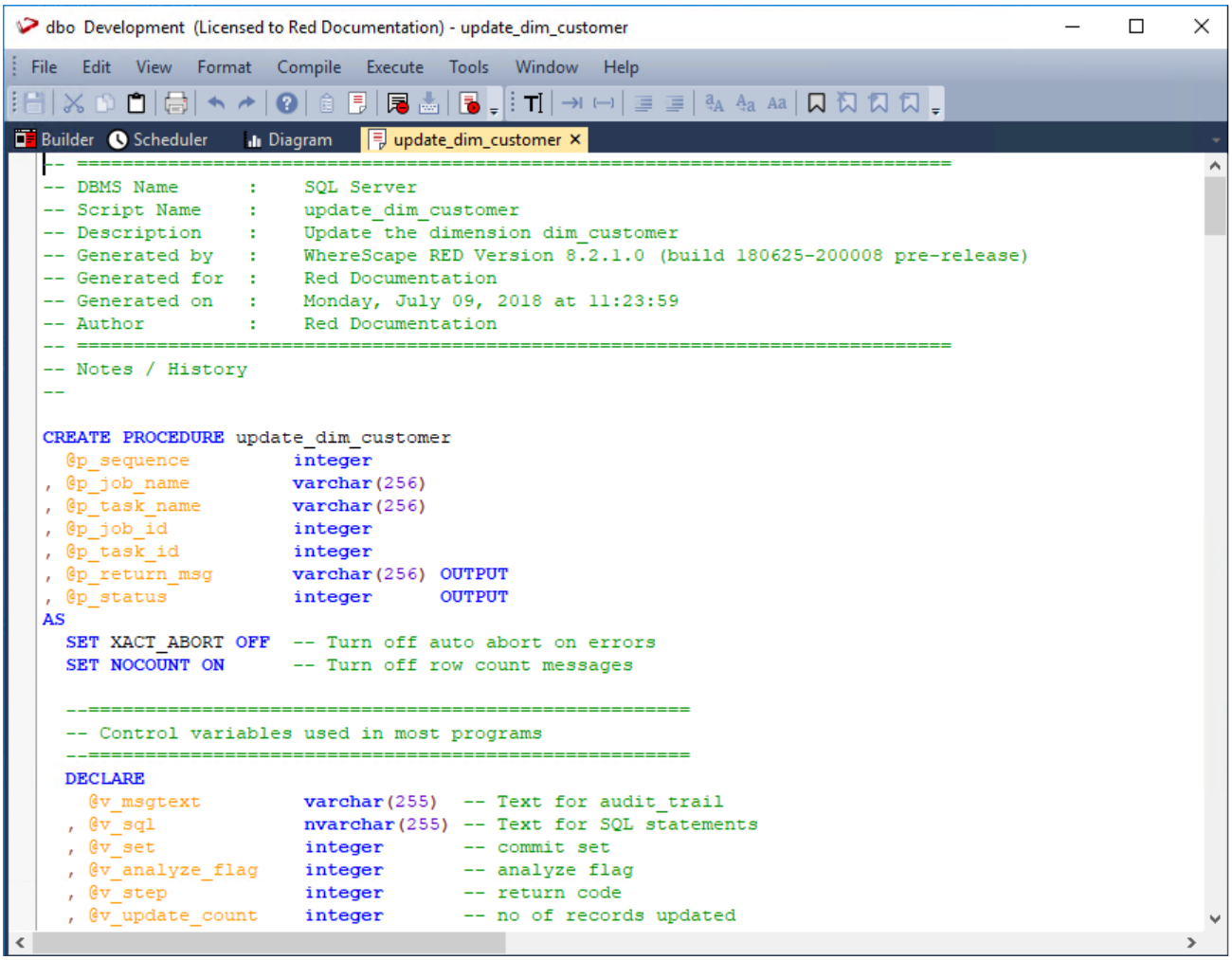
© Copyright, WhereScape Software Ltd All rights reserved.

Procedure Editor Window

The **Procedure Editor** window provides a means of viewing, editing, compiling, comparing and running procedures.

The editor includes the following functions and features:

- SQL syntax highlighting
- improved find feature (repeated find, up and down, etc.)
- toggle the display of line numbering
- a status bar that includes line number, line length, selected text length, etc.
- a context sensitive toolbar
- tab to space conversion and vice versa
- uppercase, lowercase, titlecase text conversion command
- increase/decrease test indent command
- bookmark support



Multiple editor windows can be opened at any one time, each processing a different procedure.

Comments are identified by a leading double dash (--) and displayed in green font in the window, the procedural code in black font.

The font is a fixed pitch font (by default) to make the indentation and alignment of code easier to view. The font, colors and indent size can all be changed as required.

© Copyright, WhereScape Software Ltd All rights reserved.

Toolbar

The procedure editor toolbar is shown below:

Icon	Description
	Click the **Save** button to write the procedure to the WhereScape metadata repository in the database.
	Click the **View Other Procedure/Template** button to enable the concurrent viewing of older versions of the current procedure, other procedures in the metadata, compiled procedures in the database and templates.
	Click to remove compile comments.
	Click the **Compile** button to compile the procedure. Once compiled the procedure is stored within the database as well as in the metadata. **Tip:** For Oracle data warehouses, if a procedure fails to compile it is flagged as invalid within the database.
	Click the **Execute** button to run a procedure that conforms to the WhereScape parameter syntax.
Text Editor	This toolbar provides command buttons associated with editing the procedure text.
	Click to show all white space characters.
	Click to convert space characters to tabs within the selection.
	Click to convert tab characters to spaces within the selection.
	Click to increase the indent of characters within the selection.
	Click to decrease the indent of characters within the selection.
	Click to convert all characters within the selection to uppercase.
	Click to convert all characters within the selection to lowercase.
	Click to convert all characters within the selection to title case.
	Click to toggle a bookmark on the current line.
	Click to clear all bookmarks within the current document.
	Click to move the caret to the next bookmark.
	Click to move the caret to the previous bookmark.

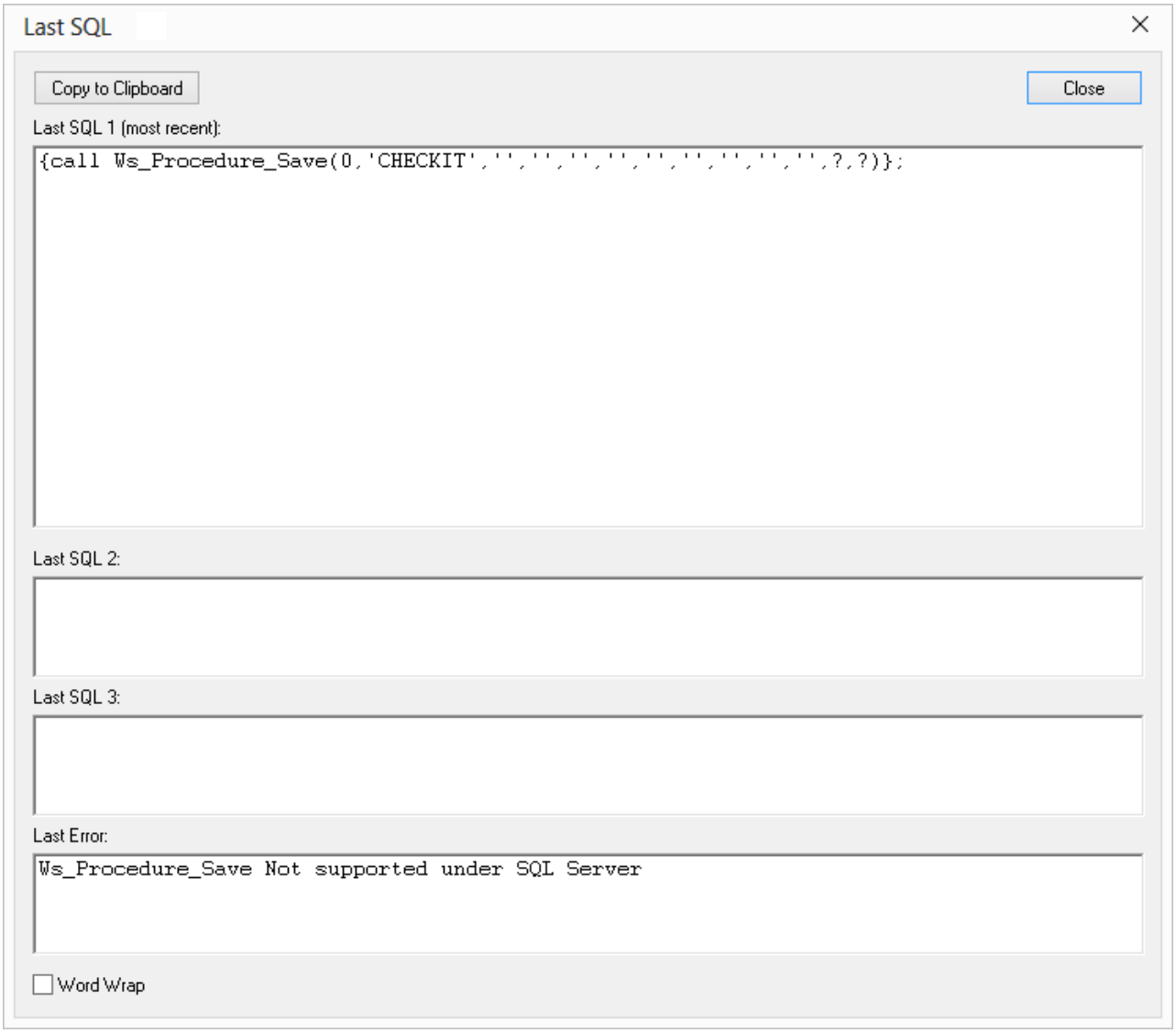
A similar editor for script editing is available. Refer to **Procedures and Scripts** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

Database Error Messages

From time to time, an SQL error may be raised by WhereScape RED. These are usually caused by the data warehouse database returning an error.
There are many causes of database error messages being returned to RED. The most common causes include permission issues and invalid SQL statements being run.
To find the exact SQL statement run by WhereScape RED, go to the **View** menu and select **Last SQL**.

The following window is displayed:



The four fields shown are:

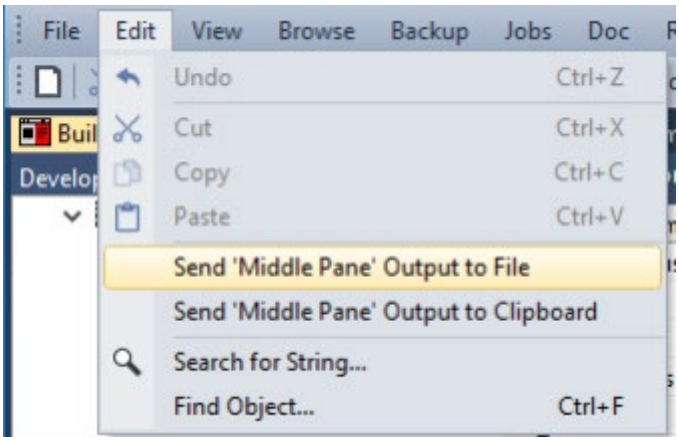
- The last SQL statement run
- The second to last SQL statement run
- The third to last SQL statement run
- The error message (if any) from the last SQL statement

In the example above, we have tried to drop a procedure that does not exist. IBM DB2 returned an error telling us RED.UPDATE_DIM_KPI does not exist in the database.

© Copyright, WhereScape Software Ltd All rights reserved.

Export Middle Pane Output

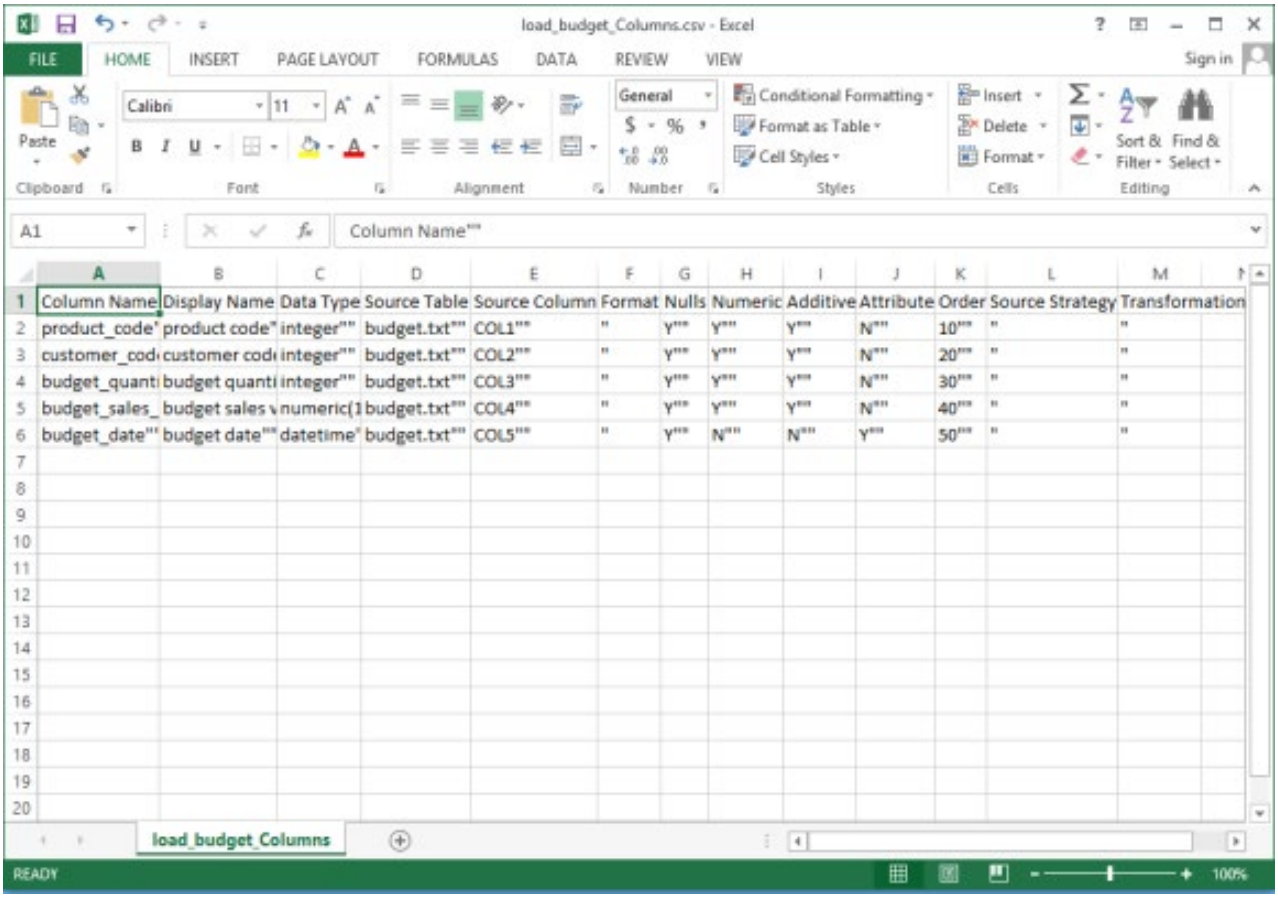
To send the middle pane output to a file, go to the **Edit** menu and select **Send 'Middle Pane' Output to File**.



A file is created in the directory defined in **Tools > User Preferences > Outputs > Output File Directory**.

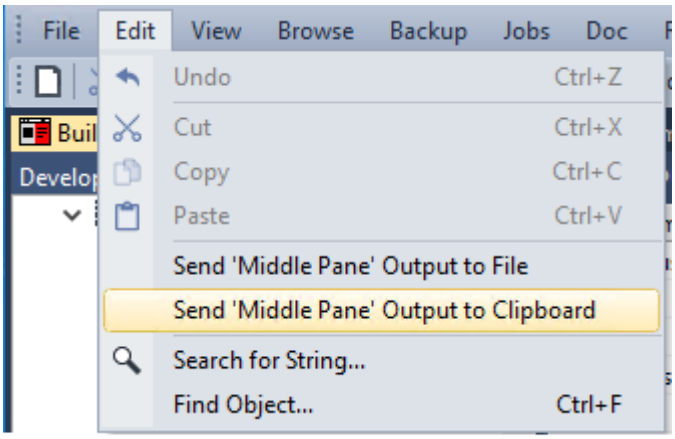
If the Output File Extension is set to .csv in **Tools > User Preferences > Outputs > Output File Extension**, then an Excel file is created.

If the file is set to auto-open in **Tools > User Preferences > Outputs > Output File Auto Open**, then the file opens automatically.



- To view the other settings for Middle Pane File Output, refer to **Export Middle Pane Output Settings** for details.
- To send the middle pane output to clipboard, go to the **Edit** menu and select **Send 'Middle Pane' Output to Clipboard**.

© Copyright, WhereScape Software Ltd All rights reserved.

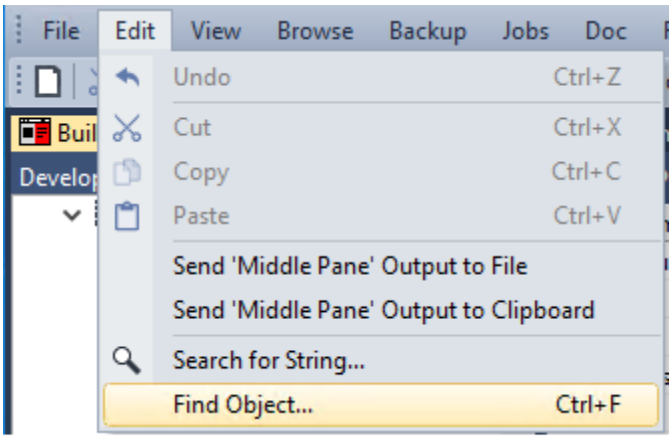


- To view the settings for **Middle Pane Clipboard Output**, refer to **Export Middle Pane Output Settings** for details.

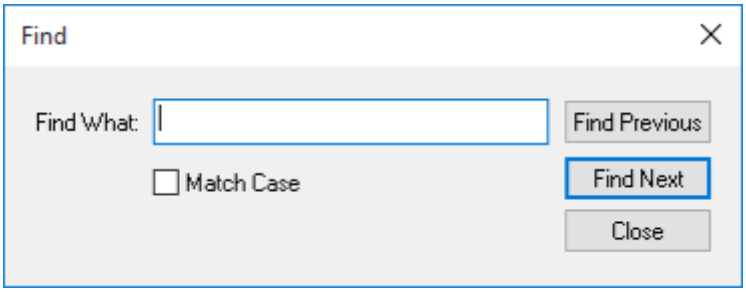
Find Function

The Find function can help users quickly find a table when the list of tables has grown to a large size. The **Find function** can be accessed two ways within WhereScape RED:

- Click anywhere in the Object Pane or Browser Pane and press **Ctrl + F**, or
- select **Edit > Find Object** (this option searches the Object Pane only).



Both of these methods open the following dialog:



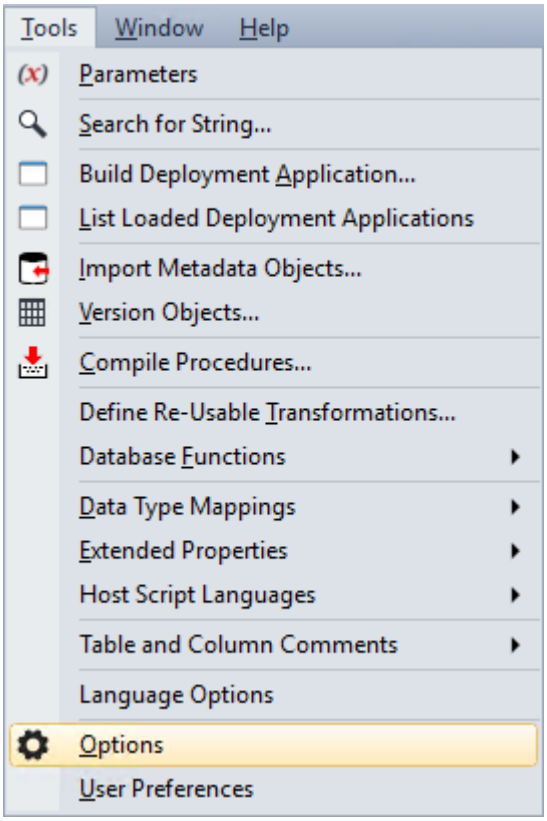
© Copyright, WhereScape Software Ltd All rights reserved.

Default Settings

This chapter describes the settings and default values that can be set for a metadata repository.
To access these settings, select the **Tools** menu and then either **Options** or **User Preferences**.

Settings - Options

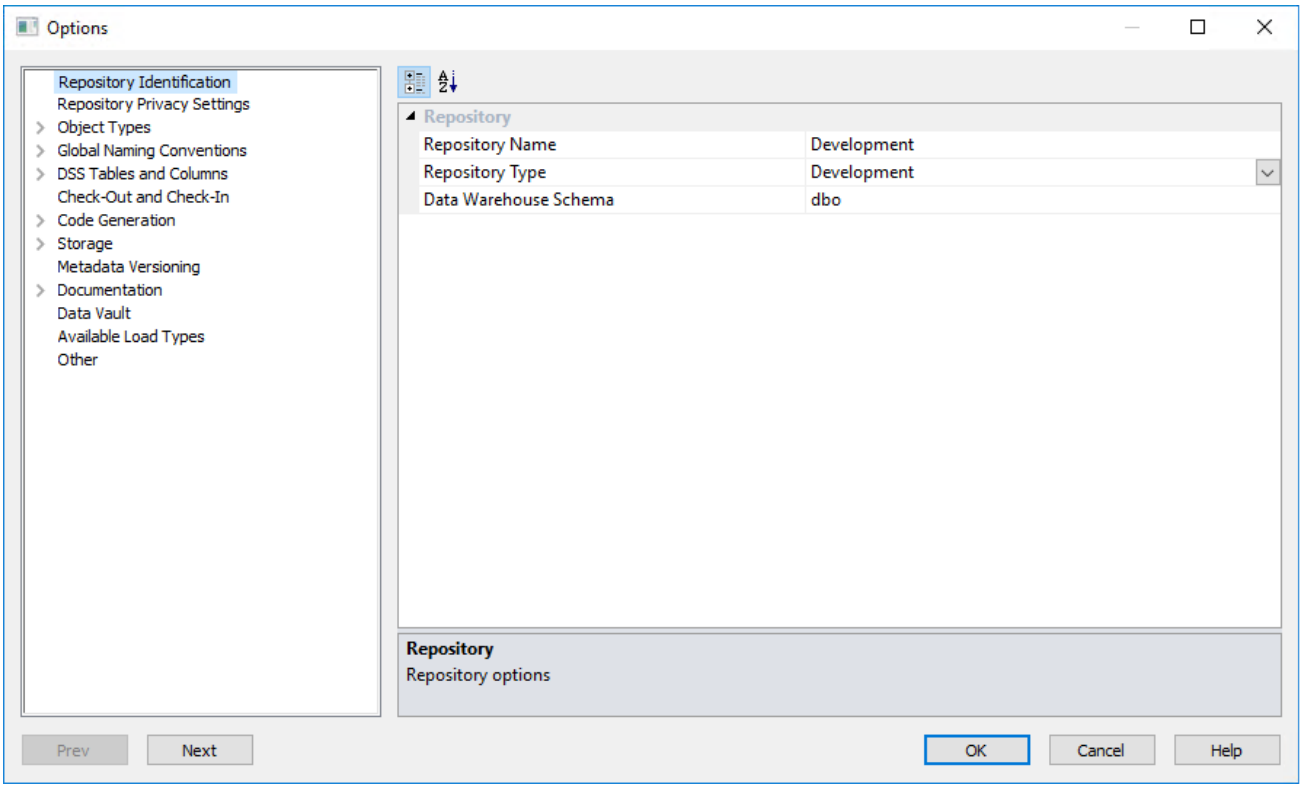
Select **Options** from the **Tools** menu.



© Copyright, WhereScape Software Ltd All rights reserved.

Repository Identification

This option enables users to set the **Repository Identification** settings.



Fields	Description
Repository Name	Set the **Name** for the repository. This name appears in the title bar in WhereScape RED . Restart WhereScape RED for repository name changes to take effect.
Repository Type	Set the **Type** for the repository. The repository type must reflect the environment. For example, a 'Production' type must be chosen for the production environment.
Data Warehouse Schema	Set the **Schema** for the repository. The schema of the data warehouse is used by the WhereScape RED scheduler. You should not normally need to change this value.

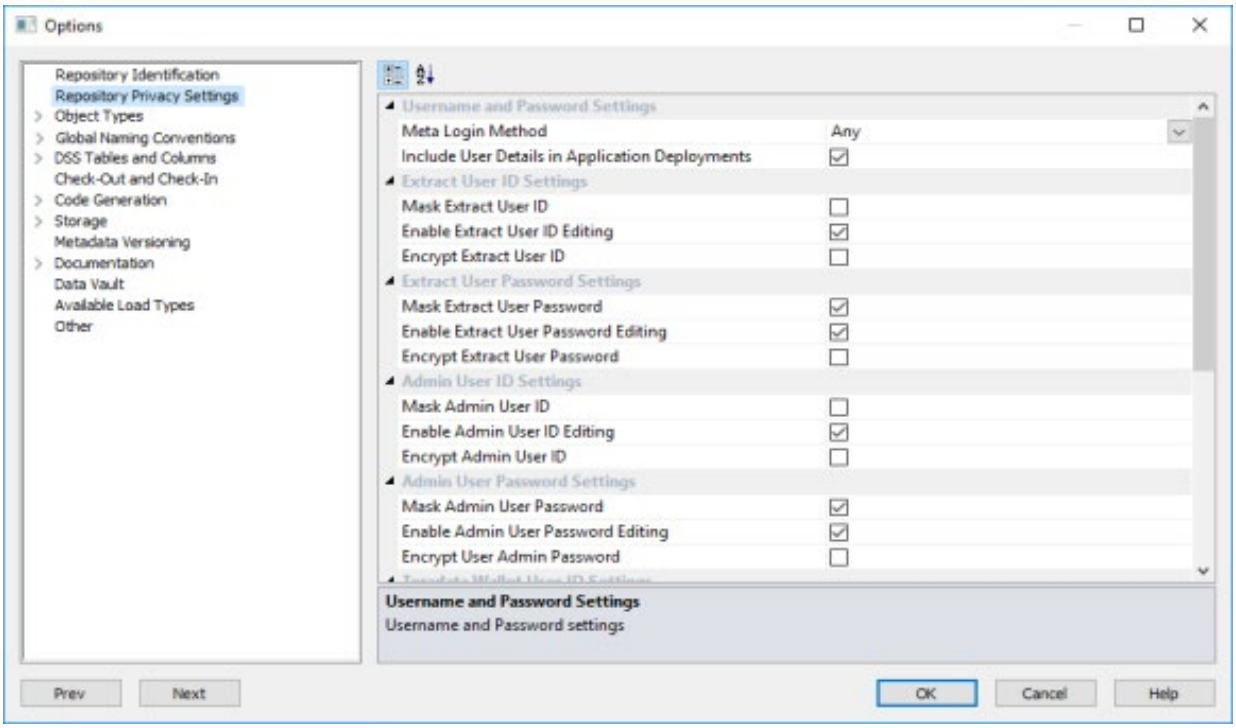
© Copyright, WhereScape Software Ltd All rights reserved.

Repository Privacy Settings

This option enables users to set the **Repository Privacy Settings**.

Warning

For UNIX/Linux scheduler processing, the **Encrypt** User and Password options cannot be used. Encrypt options are only supported when using a Windows scheduler.



Changing Repository Settings

Since the repository privacy settings can be configured from the **Tools > Options** menu, for an environment to be secured, **a database administrator has to change the permissions on table ws_meta_admin table to read-only after the appropriate repository privacy change settings** in WhereScape RED have been made.

Note:

Changing this set of permissions to read-only is something which occurs outside of WhereScape RED and is dependent on the specific meta data database.

Username and Password Settings

Fields	Description
Meta Login Method	This option can be set to restrict users from using a particular login method for the meta repository.
Include User Details in Application Deployments	Includes or excludes User Details in Application Deployment packages.

Extract User ID Settings

Fields	Description
Mask Extract User ID	Masks the input of the "Extract/Unix/Windows User ID" on the connection properties.

© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
Enable Extract User ID Editing	Enables editing the "Extract/Unix/Windows User ID" via the connection properties.
Encrypt Extract User ID	Encrypts "Extract/Unix/Windows User ID" in the meta repository using WhereScape encryption.

Extract User ID Settings

Fields	Description
Mask Extract User Password	Masks the input of the "Extract/Unix/Windows User Password" on the connection properties.
Enable Extract User Password Editing	Enables editing "Extract/Unix/Windows User Password" via the connection properties.
Encrypt Extract User Password	Encrypts "Extract/Unix/Windows User Password" in the meta repository using WhereScape encryption.

Admin User ID Settings

Fields	Description
Mask Admin User ID	Masks the input of the "Admin/DSS User ID" on the connection properties.
Enable Admin User ID Editing	Enables editing the "Admin/DSS User ID" via the connection properties.
Encrypt Admin User ID	Encrypts "Admin/DSS User ID" in the meta repository using WhereScape encryption.

Admin User Password Settings

Fields	Description
Mask Admin User Password	Masks the input of the "Admin/DSS Password" on the connection properties.
Enable Admin Password Editing	Enables editing the "Admin/DSS Password" via the connection properties.
Encrypt Admin Password	Encrypts "Admin/DSS Password" in the meta repository using RED encryption.

Teradata Wallet User ID Settings

Fields	Description
Mask Teradata Wallet User ID	Masks the input of the "Teradata Wallet User ID" on the connection properties
Enable Teradata Wallet User ID Editing	Enables editing the "Teradata Wallet User ID" via the connection properties.
Encrypt Teradata Wallet User ID	Encrypts the "Teradata Wallet User ID" in the meta repository using WhereScape encryption.

Teradata Wallet String Settings

Fields	Description
Mask Teradata Wallet String	Masks the input of the "Teradata Wallet String" on the connection properties.
Enable Teradata Wallet String Editing	Enables editing the "Teradata Wallet String" via the connection properties.

© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
Encrypt Teradata Wallet String	Encrypts the "Teradata Wallet String" in the meta repository using WhereScape encryption.

JDBC User ID Settings

Fields	Description
Mask JDBC User ID	Masks the input of the "JDBC User ID" on the connection properties.
Enable JDBC User ID Editing	Enables editing the "JDBC User ID" via the connection properties.

JDBC User Password Settings

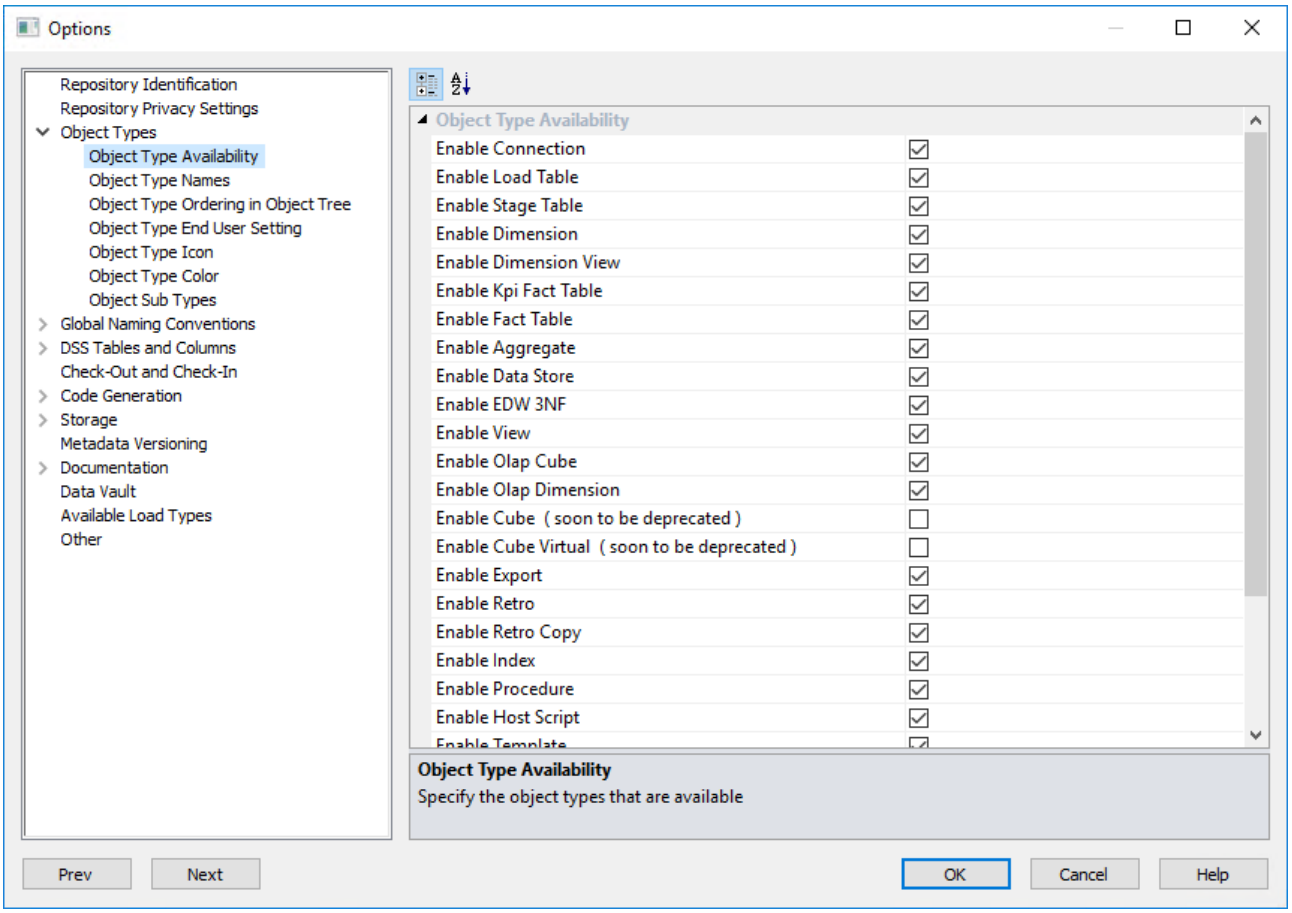
Fields	Description
Mask JDBC User Password	Masks the input of the "JDBC User Password" on the connection properties.
Enable JDBC User Password Editing	Enables editing the "JDBC User Password" via the connection properties.

© Copyright, WhereScape Software Ltd All rights reserved.

Object Types

Object Type Availability

This option enables you to activate or deactivate the various object types within the data warehouse repository.
Enable/Disable object types in the data warehouse by selecting/clearing the availability check-boxes for each object type. All settings are enabled by default, except for the **Enable Cube** and **Enable Virtual Cube** options which are soon to be deprecated.



Object Type Names

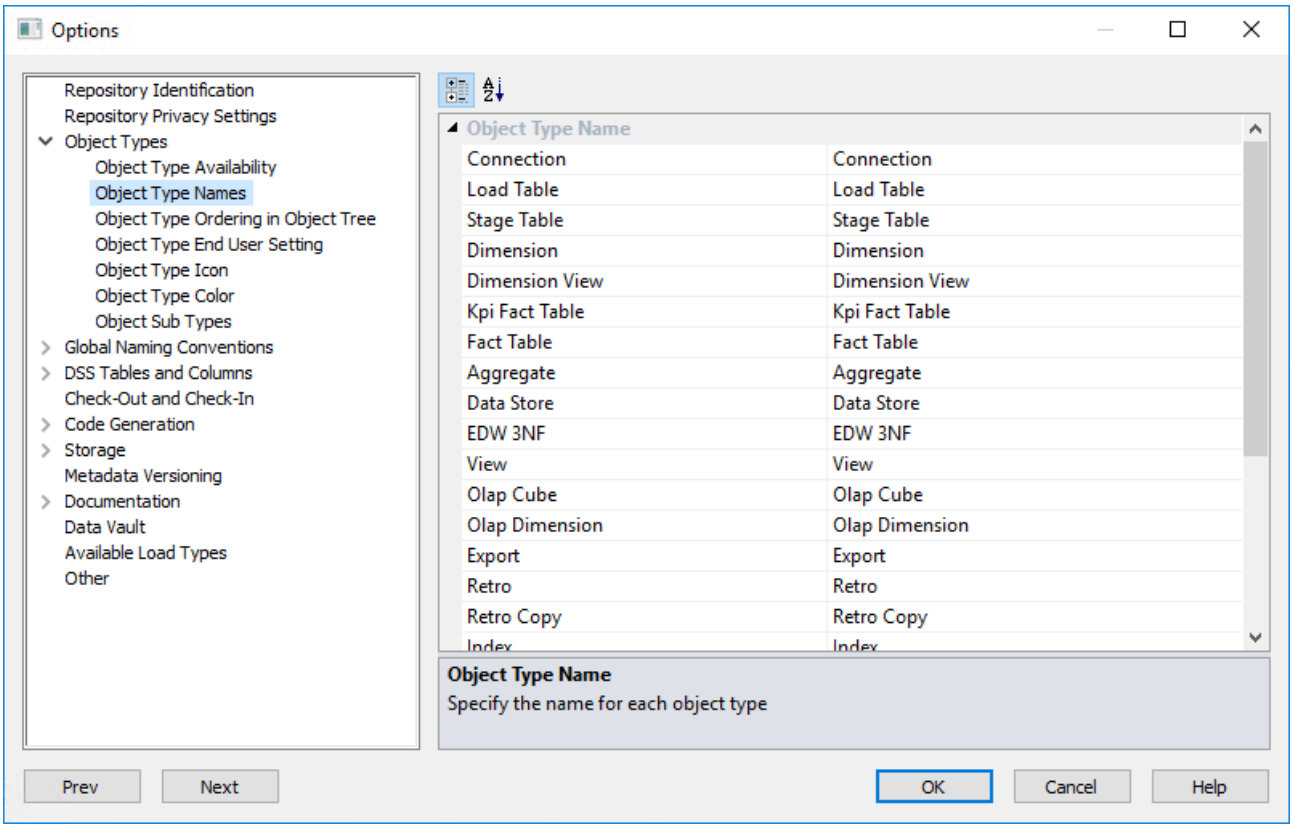
This option enables you to set the **names** for the various object types.

Set the **Name** for each object type.

Note:

Data Vault Repository Types: Users with Data Vault model type licenses that chose a **Data Vault repository type** while creating the RED metadata repository will have appropriate Data Vault repository default settings, such as Object Type Names, Global Naming of Tables, Indexes, Key Columns and Procedures/Scripts, as well as other repository settings/user preferences. These repository types will have their default Object Type Names of Normalized and Data Store objects set to Hub/Link and Satellite.

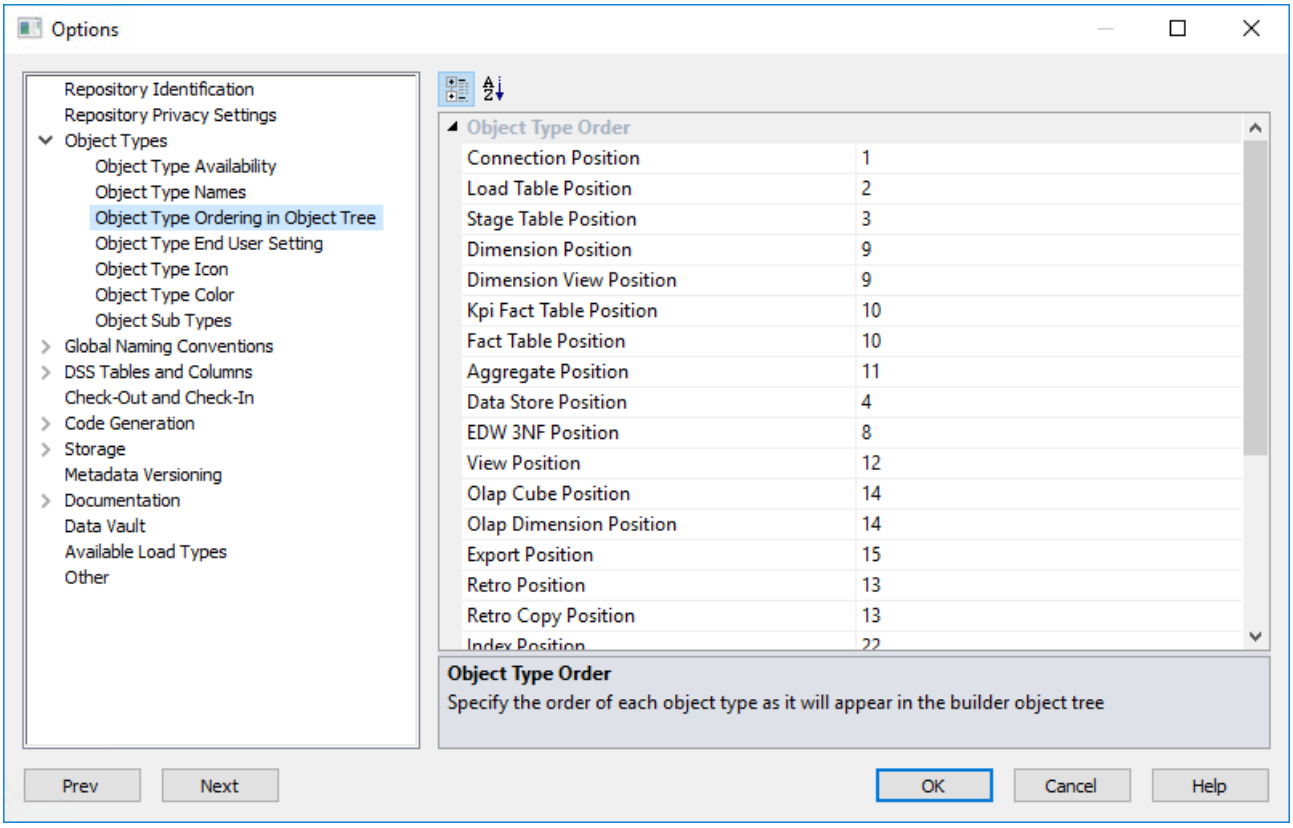
© Copyright, WhereScape Software Ltd All rights reserved.



Object Type Ordering

This option enables you to set the **ordering** in which the object types appear in the object tree.

Set the **Ordering** of the object types as displayed in the object tree pane.



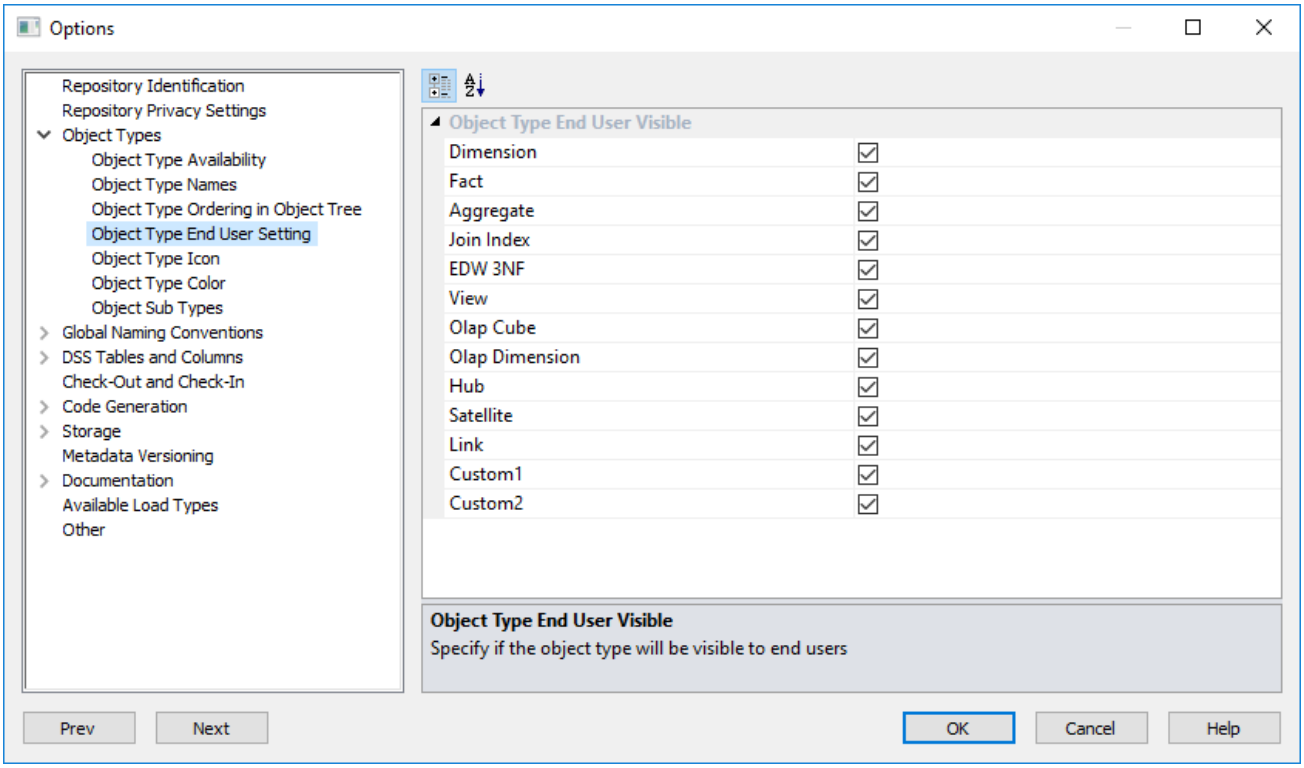
© Copyright, WhereScape Software Ltd All rights reserved.

Object Type End User Setting

This option enables you to set the Object types as **end user objects**.

All the options in this screen are selected by default.

Clear an object type check box to remove it as an end user object, else leave selected.

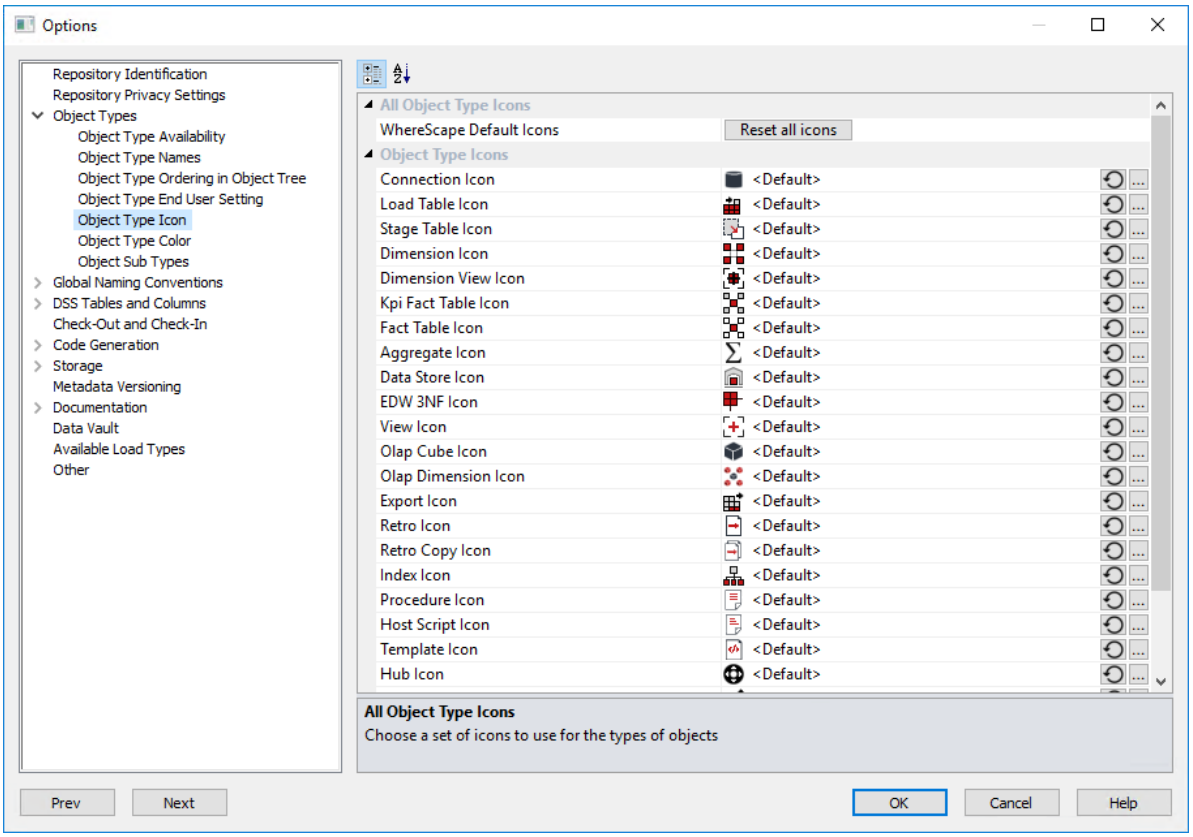


© Copyright, WhereScape Software Ltd All rights reserved.

Object Type Icon

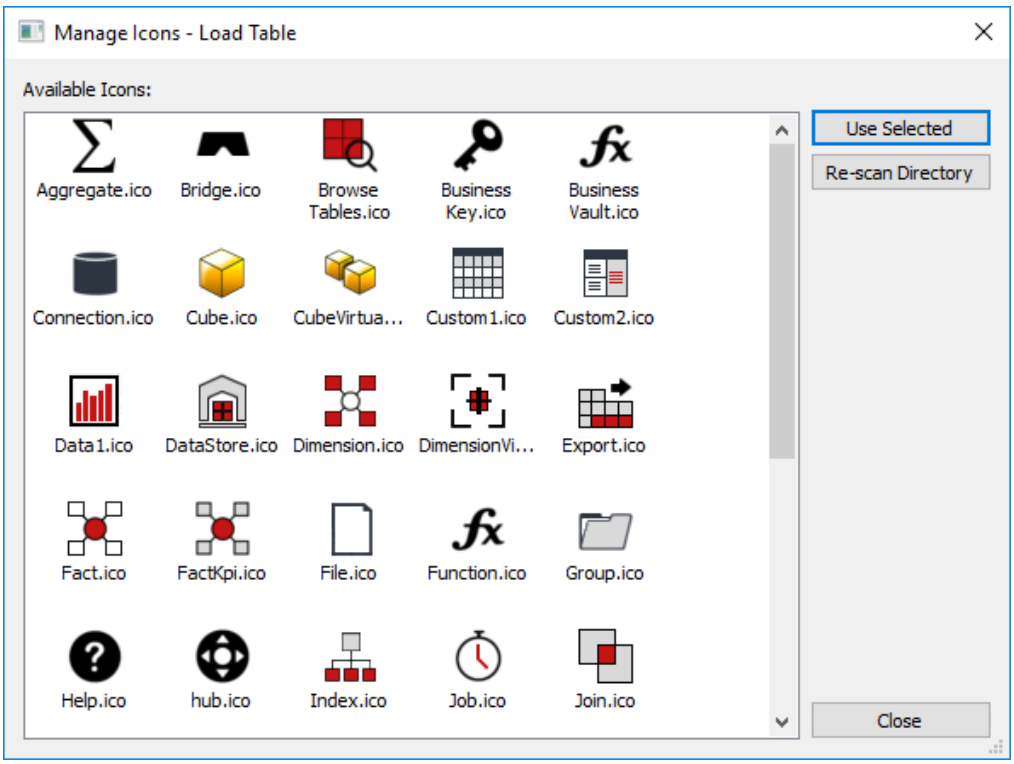
This option enables you to configure the **Icons** for all Object Types. To configure custom Object Type icons:

1. Create an 'Icons' folder in the WhereScape RED install directory, if it doesn't exist. For example:
   C:\Program Files (x86)\WhereScape\Icons
2. Place custom '**.ico**' files in the Icons folder.
3. In RED, select **Options > Object Types > Object Type Icon**.

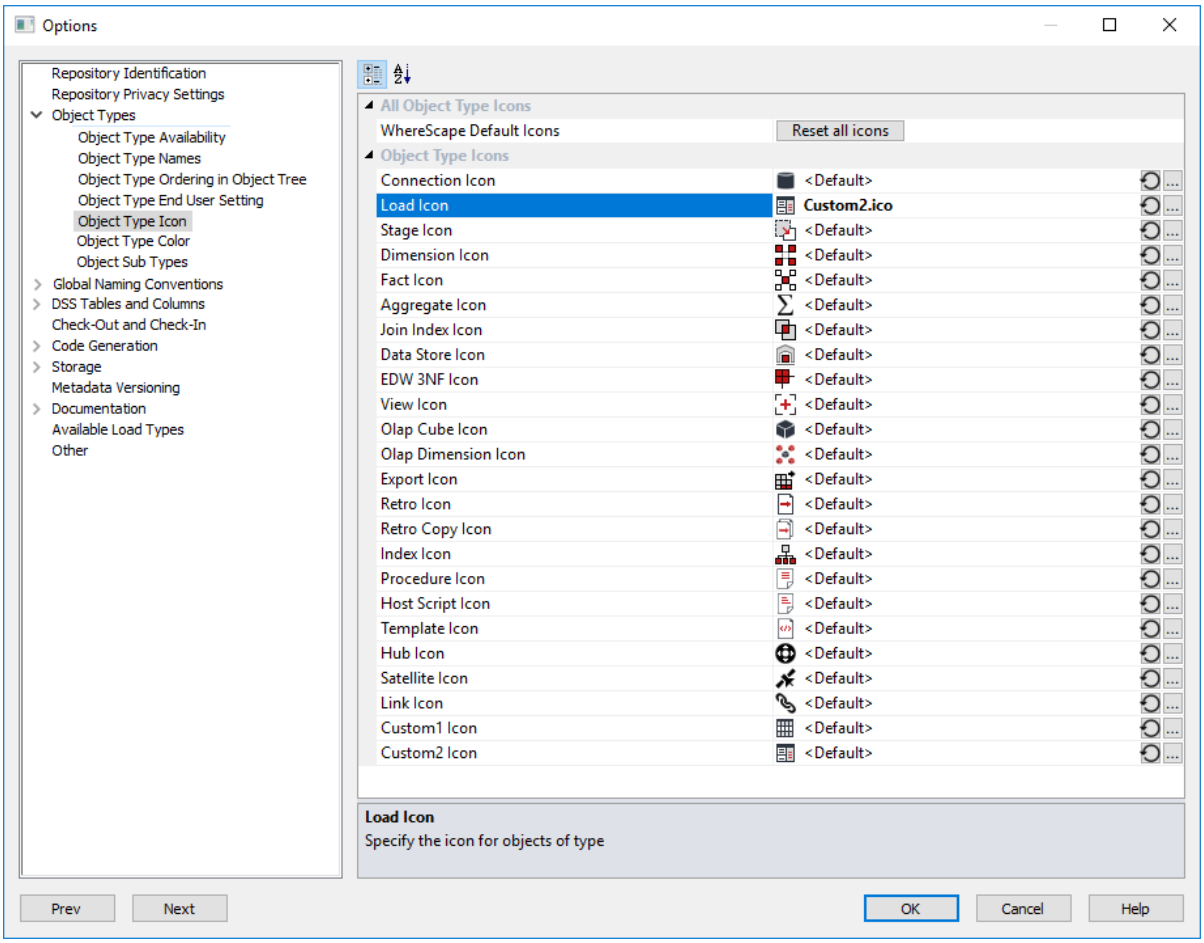


© Copyright, WhereScape Software Ltd All rights reserved.

4. Click the ellipses button **'...'** next to each Object Type and select the desired icon.



5. All configured icons are displayed as file names in the screen. To save changes, click **OK**.



© Copyright, WhereScape Software Ltd All rights reserved.

To Reset An Icon

To reset an icon to default, click the reset button next to the icon.

Reset All Icons

To reset all icons to default, click the **Reset All Icons** button at the top of the dialog.

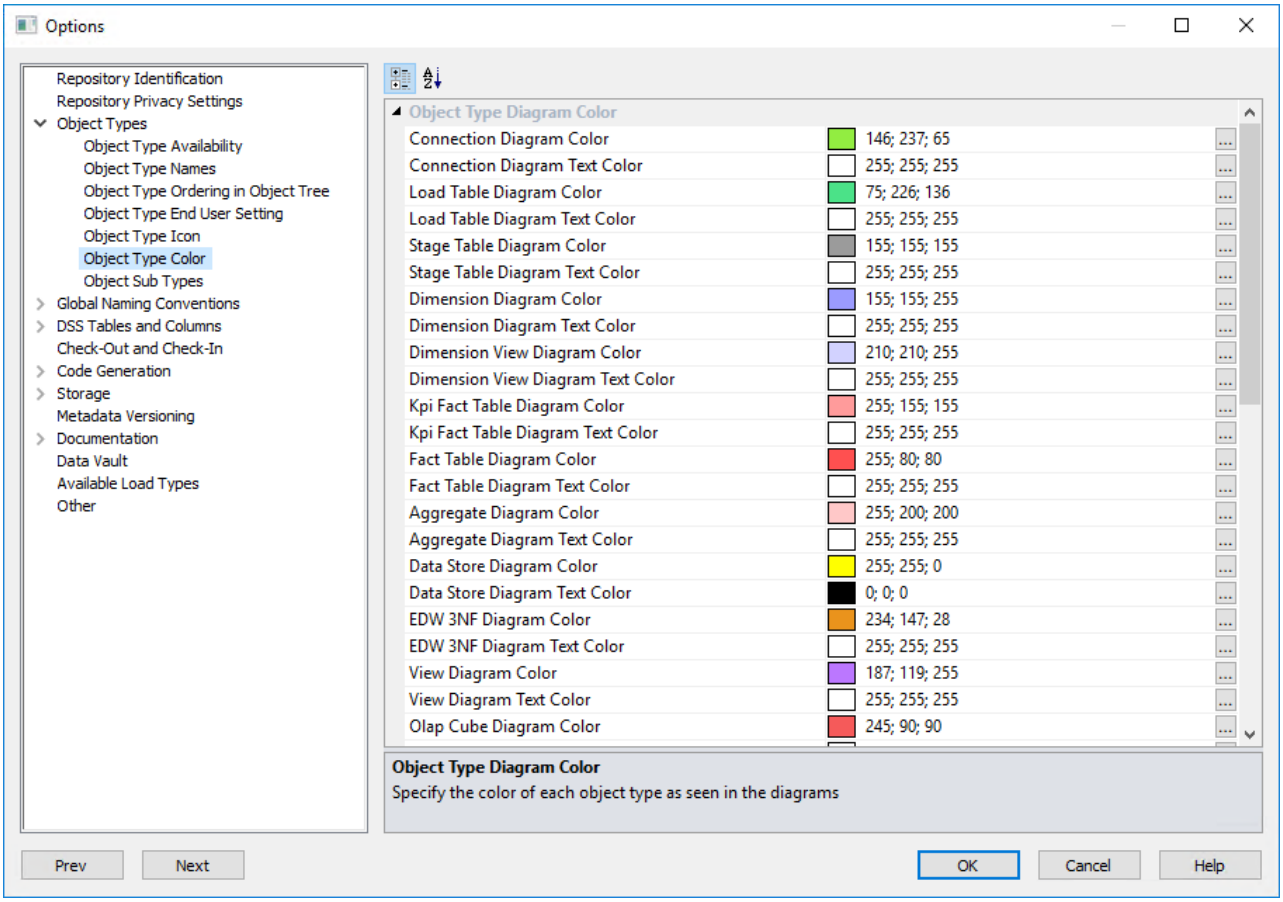
Note:

All installations must have a copy of the icon directory.

Object Type Color

This option enables you to set the **diagram colors** for each object type.

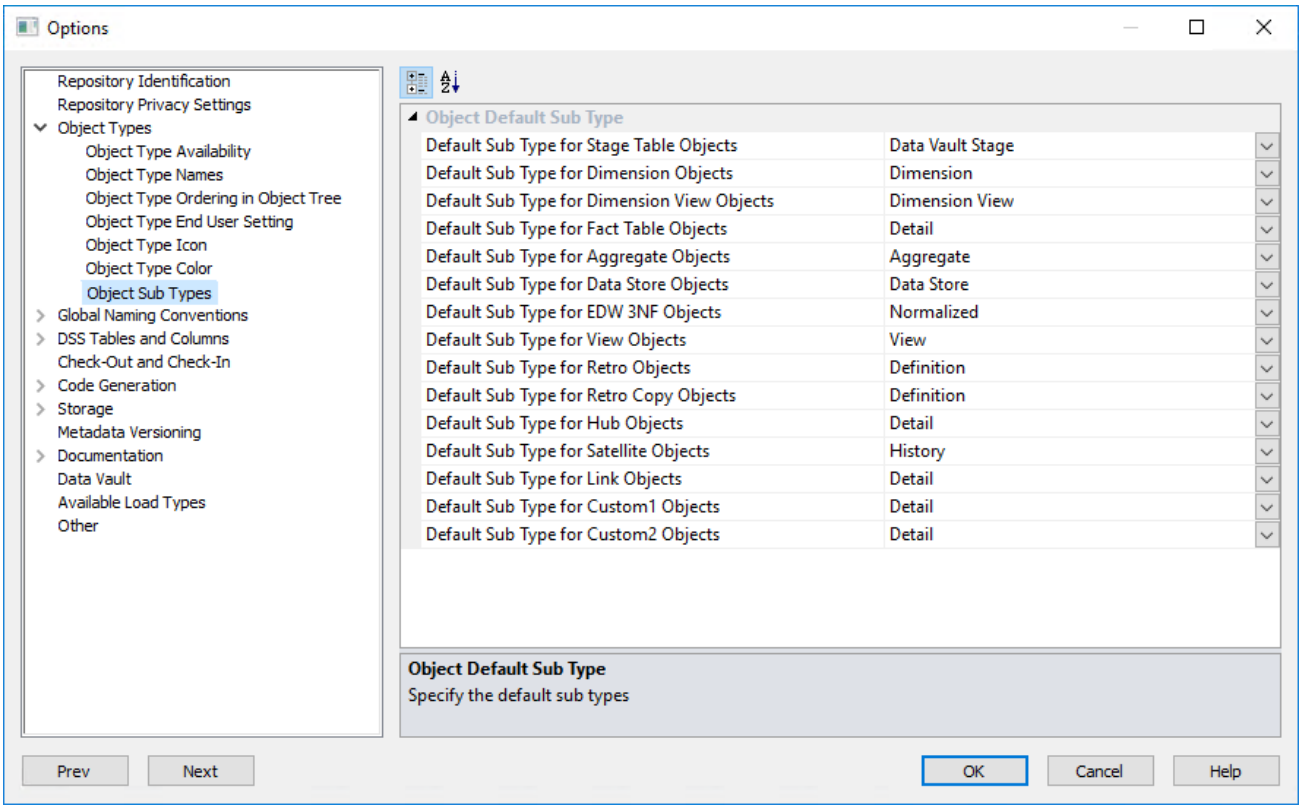
Set the **Diagram Color** for each object type.



© Copyright, WhereScape Software Ltd All rights reserved.

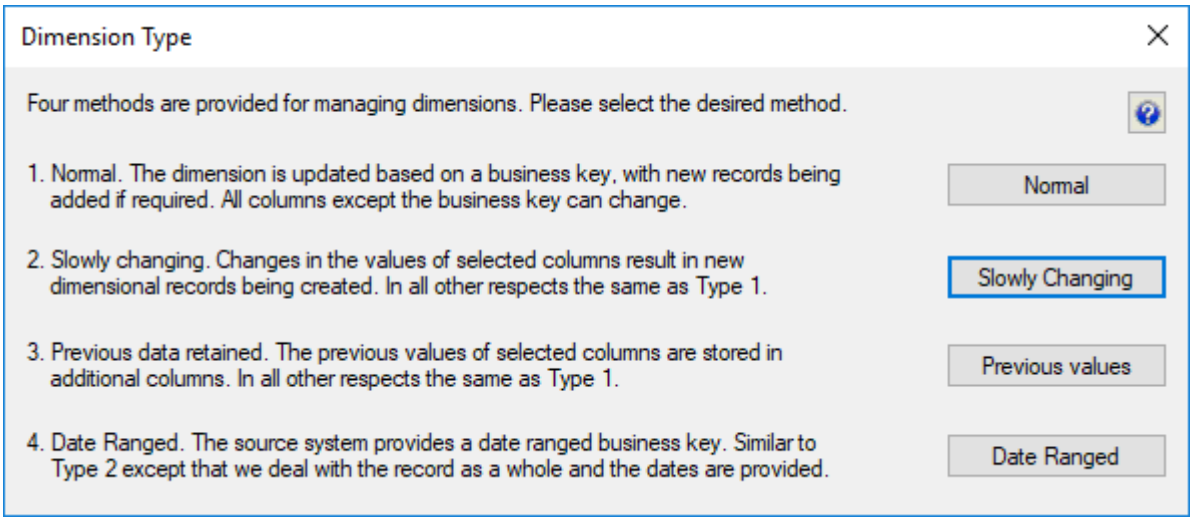
Object Sub Types

This option enables you to set the **Default Sub Type** for enabled object types in RED. Select the desired default sub-types from the Object's drop-down lists.



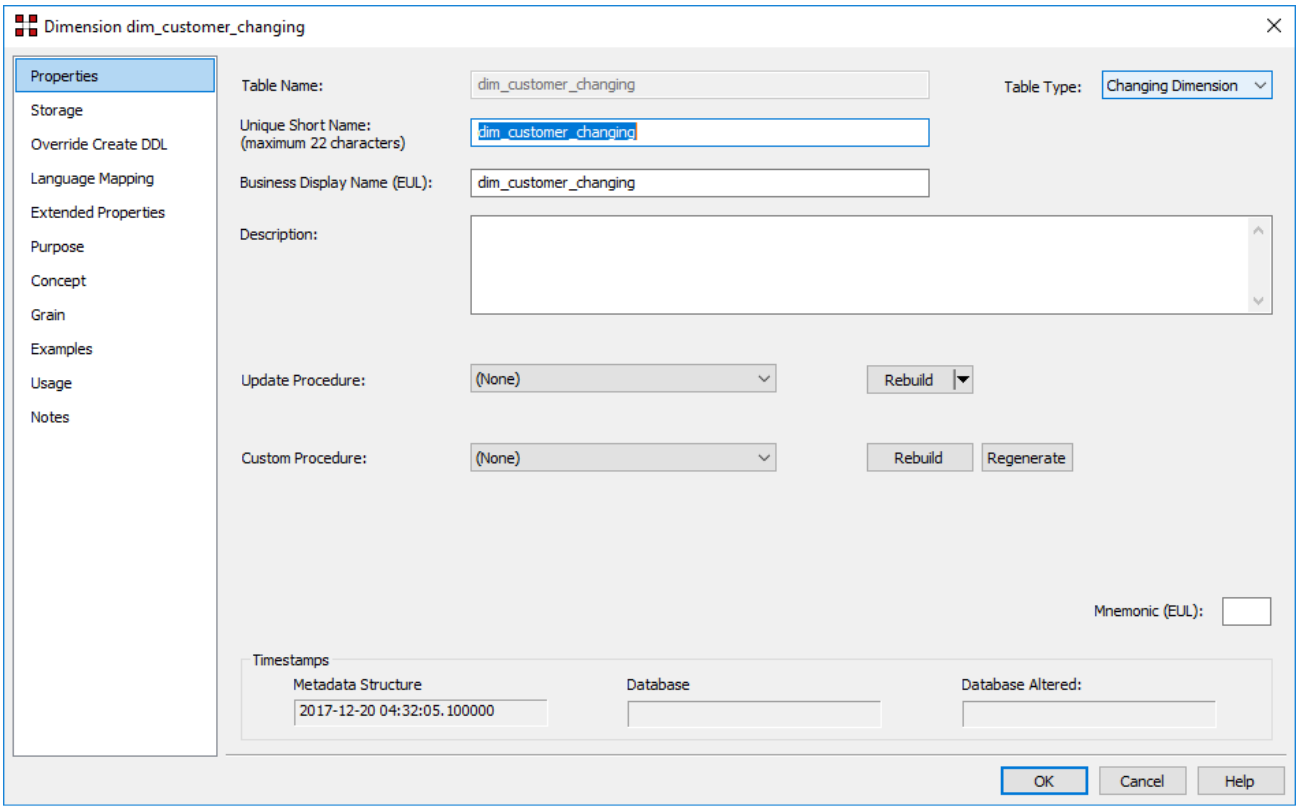
For example, to have **Dimension** objects created in RED as **Changing Dimensions** at the time of drag and drop, select the **Changing Dimension** option in the **Default Sub Type for Dimension Objects**.

After the table is dragged and dropped, users can simply hit enter to proceed on the Dimension Type where the **Slowly Changing** type is already the default sub type option, as previously selected in **Tool > Options** menu.



© Copyright, WhereScape Software Ltd All rights reserved.

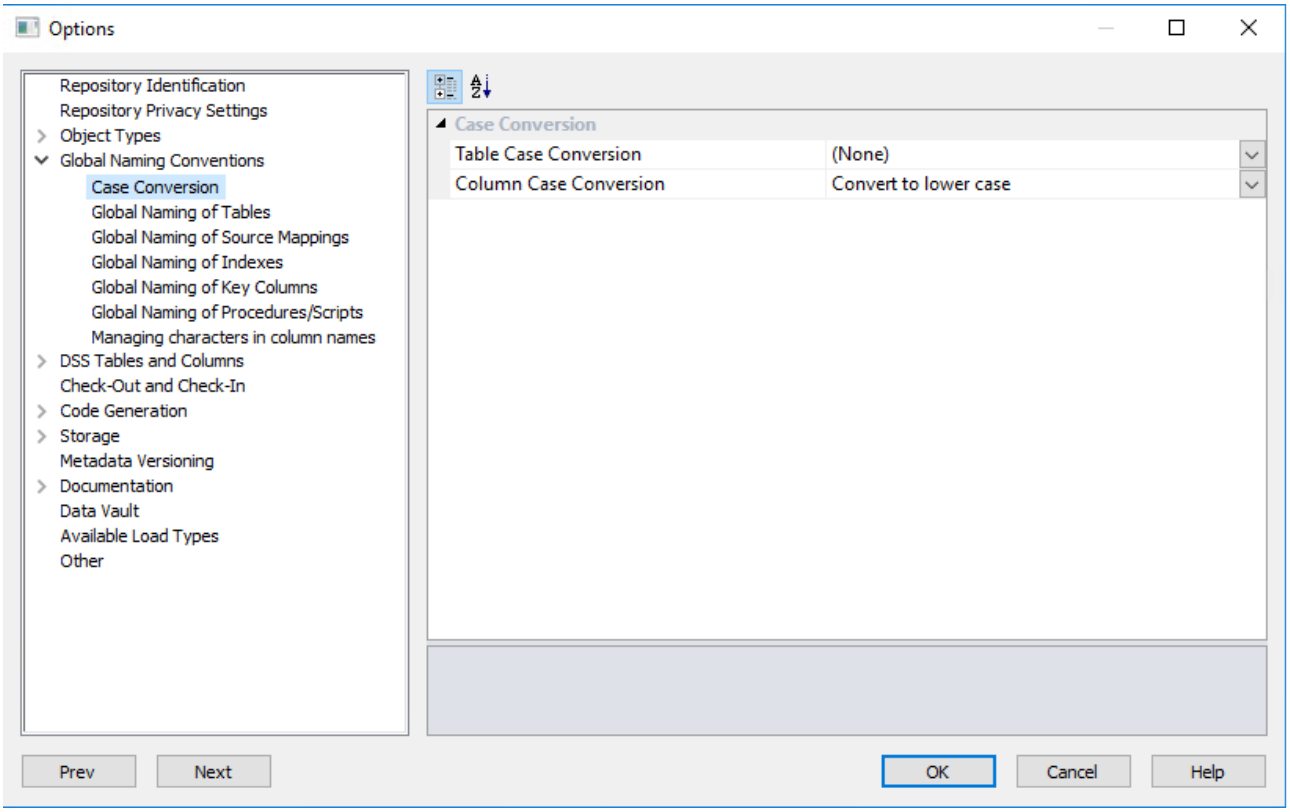
The **Dimension Properties** screen also reflects the selected table sub type on the **Table Type** drop-down list.



Global Naming Conventions

Case Conversion

This option enables you to set the **Case Conversion** methods for the tables and columns in RED.



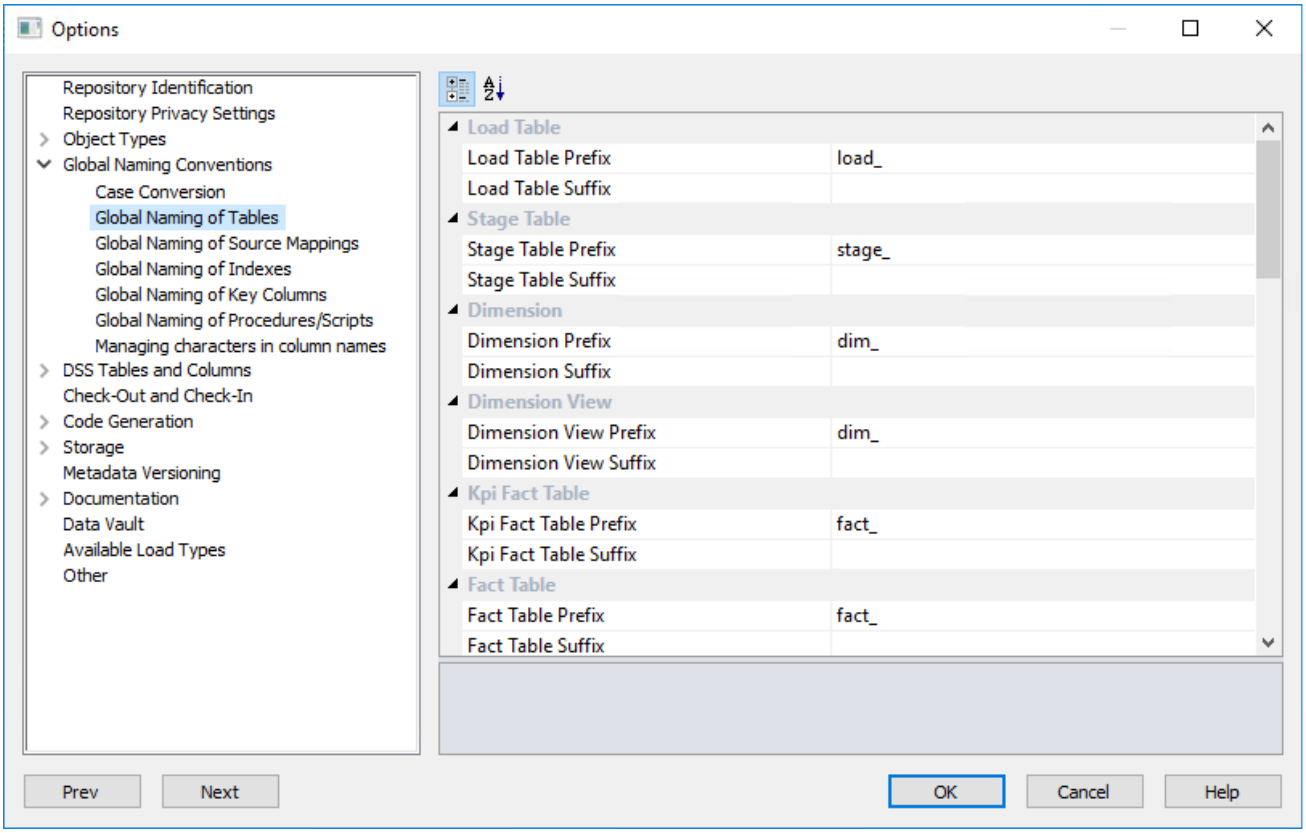
© Copyright, WhereScape Software Ltd All rights reserved.

Case Conversion

Set the **Table Case Conversion** method and the **Column Case Conversion** method from the drop-down lists.

Global Naming of Tables

This option enables you to set the **Global Naming of Tables** options.



A prefix and/or a suffix string can be applied to an object name. Within Oracle and IBM DB2, a table name may be a maximum of 30 characters long, so these pre and post fix strings should not be more than eight characters long (WhereScape RED short names are a maximum of 22 long in Oracle and SQL Server and 12 long in DB2).

From the example screen above, if a source table called **customer** (with a short name of 'customer') was dragged into a Load table drop target then the default name would be **load_customer**.

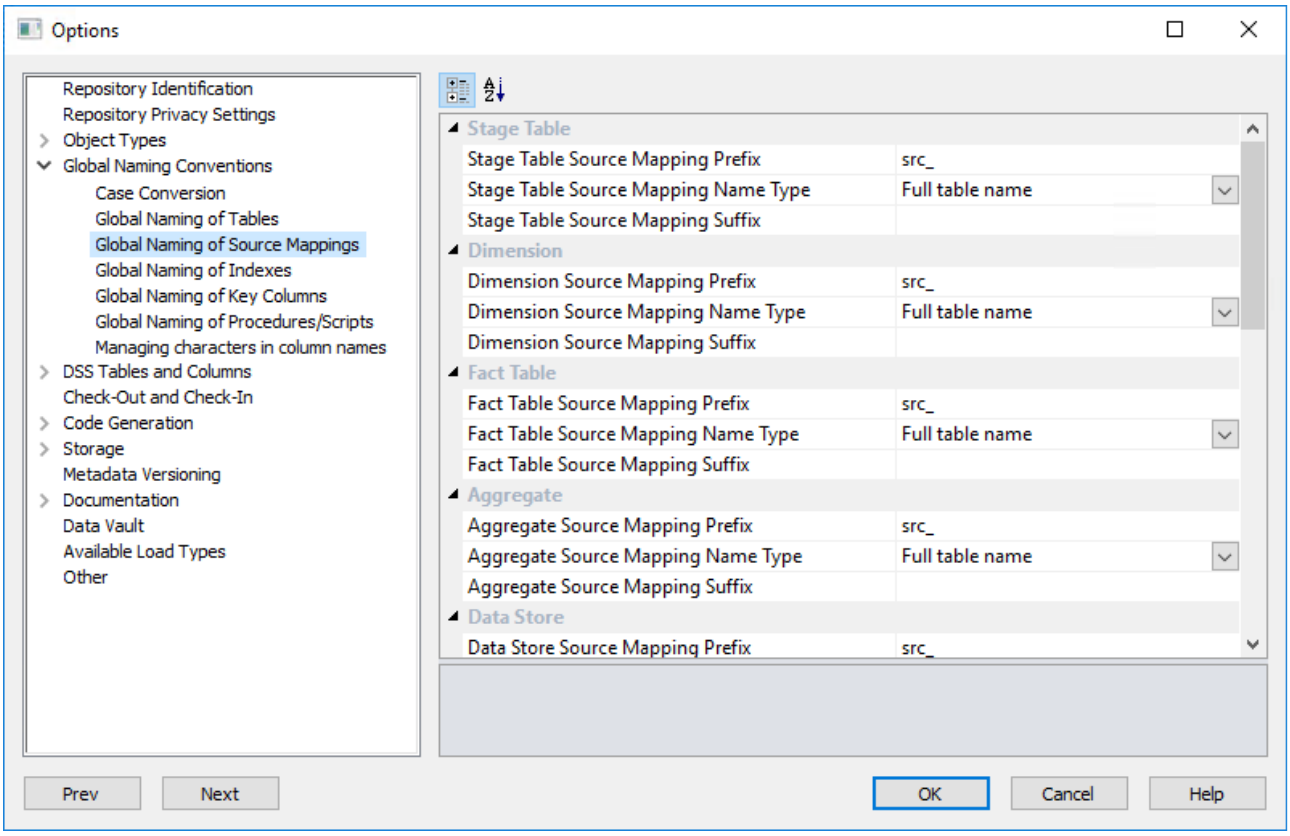
The object name defaults shown above are the values that are installed with the base metadata.

They can be changed at any stage; however, the change does not affect any existing objects. Therefore, if a new naming regime is chosen, any existing objects need to be renamed through the **Properties** screen of the object.

© Copyright, WhereScape Software Ltd All rights reserved.

Global Naming of Source Mappings

This option enables you to set the **Global Naming of Source Mappings** options.



Similar to table objects created in RED, a prefix and/or a suffix string can also be applied to source mapping child objects.

From the example screen above, if a source table was dragged into a table drop target called **stage_customer** (with a full table name of 'customer') then the default name for the first source mapping object created would be **src_stage_customer_1**.

The same applies for the second source mapping object created—the object would be named **src_stage_customer_2**.

The object name defaults shown above are the values that are installed with the base metadata.

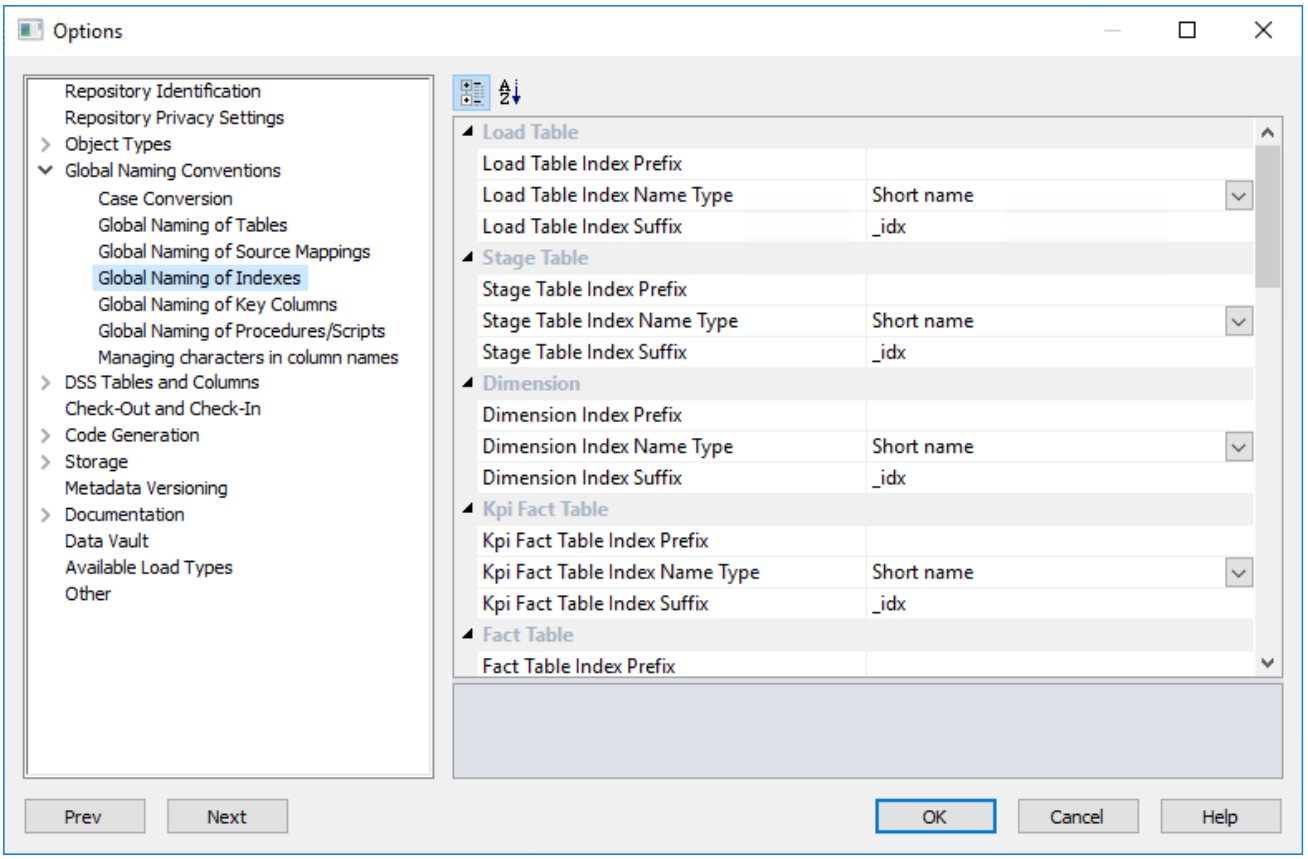
They can be changed at any stage; however, the change does not affect any existing objects. Therefore, if a new naming regime is chosen, any existing objects has to be renamed through the **Properties** screen of the object.

Source mappings don't result in the creation of tables, so the length of their name is not constrained by the table object name length limited applied by Oracle and IBM DB2 (30 characters). However, the source mapping name is used as the basis for the default procedure name and procedure names do have a 30 character length limited in these two database technologies.

© Copyright, WhereScape Software Ltd All rights reserved.

Global Naming of Indexes

This option enables you to set the **Global Naming of Indexes** options.



Whenever a new procedure is defined, WhereScape RED builds or rebuilds a standard set of indexes for the table. These indexes are created using the standard defined. As with the key naming, we can set either a prefix or a suffix value, or in fact both, as well as choosing the use of either the table name or the short name associated with the table.

In addition to the naming specifications above, WhereScape RED adds up to a further 3 characters to the end of the index name. These additional values will be "_0" through "_99", or "_A" through "_Z", or "_SC".

When a new index is manually added it will have the additional value of "_x" by default. This should be changed. The WhereScape RED naming standard for indexes is described below, but any valid name may be used.

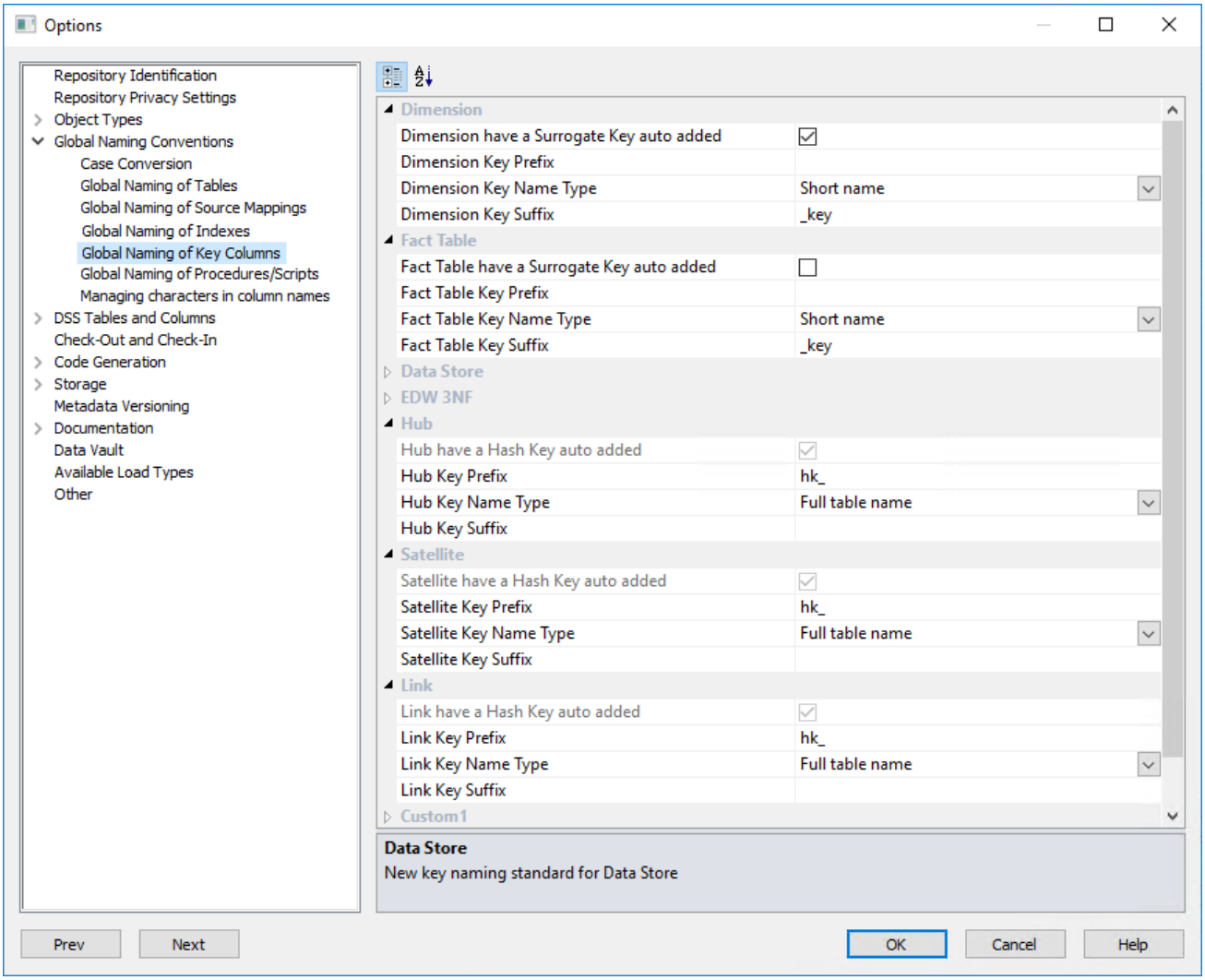
From the example screen above, a fact table would have indexes generated using the short name and with a suffix of "_idx". Therefore, a fact_sales fact table would have indexes, such as fact_sales_idx_x.

Ultimate suffix	Meaning
_0	artificial key
_1 thru _99	bitmap key index on dimensional join
_A	primary business key
_B thru _Z	secondary business keys
_SC	key to support slowly changing dimensions

© Copyright, WhereScape Software Ltd All rights reserved.

Global Naming of Key Columns

This option enables you to set the **Global Naming of Key Columns**.



During the drag and drop generation of new dimension and fact tables, WhereScape RED builds an artificial (surrogate) key for the table.

The naming convention for that key can be set through the same menu option as above. As well as potential pre-fix and suffix values, you also need to choose between the inclusion of the full table name, or the short name assigned to each table.

In the example screen above (which is the default), a dimension table key would use the table **short name** and have a suffix of "**_key**". Therefore, a load_customer table example would generate a key called **dim_customer_key**, if it was dragged into a dimension drop target.

The example above displays the defaults for Dimension options, to set these fields on Fact, Data Store and EDW 3NF tables, etc. expand the fields below Dimension to view and set your required options.

Fields	Description
Dimension have a Surrogate Key auto added	Select this option if a Surrogate key column is to be added automatically to a table. Default for Dimension is selected. Default for Fact, Data Store and EDW 3NF is not selected.
Dimension Key Prefix	Key prefix that can be added to a new Dimension Key.
Dimension Key Name Type	Key name type for new Dimension keys. Select between Short name, Full table name and Base name.
Dimension Key Suffix	Key suffix that can be added to a new Dimension key.

© Copyright, WhereScape Software Ltd All rights reserved.

Note:

For Data Vault table objects, hash keys are added automatically by the **Data Vault stage table creation wizard** and are required when using WhereScape data vault code generation templates. Therefore, the **Hash key auto added** option for Hub, Satellite and Link tables are enabled by default and cannot be modified. Refer to **Data Vault Settings** for details.

Global Naming of Procedures/Scripts

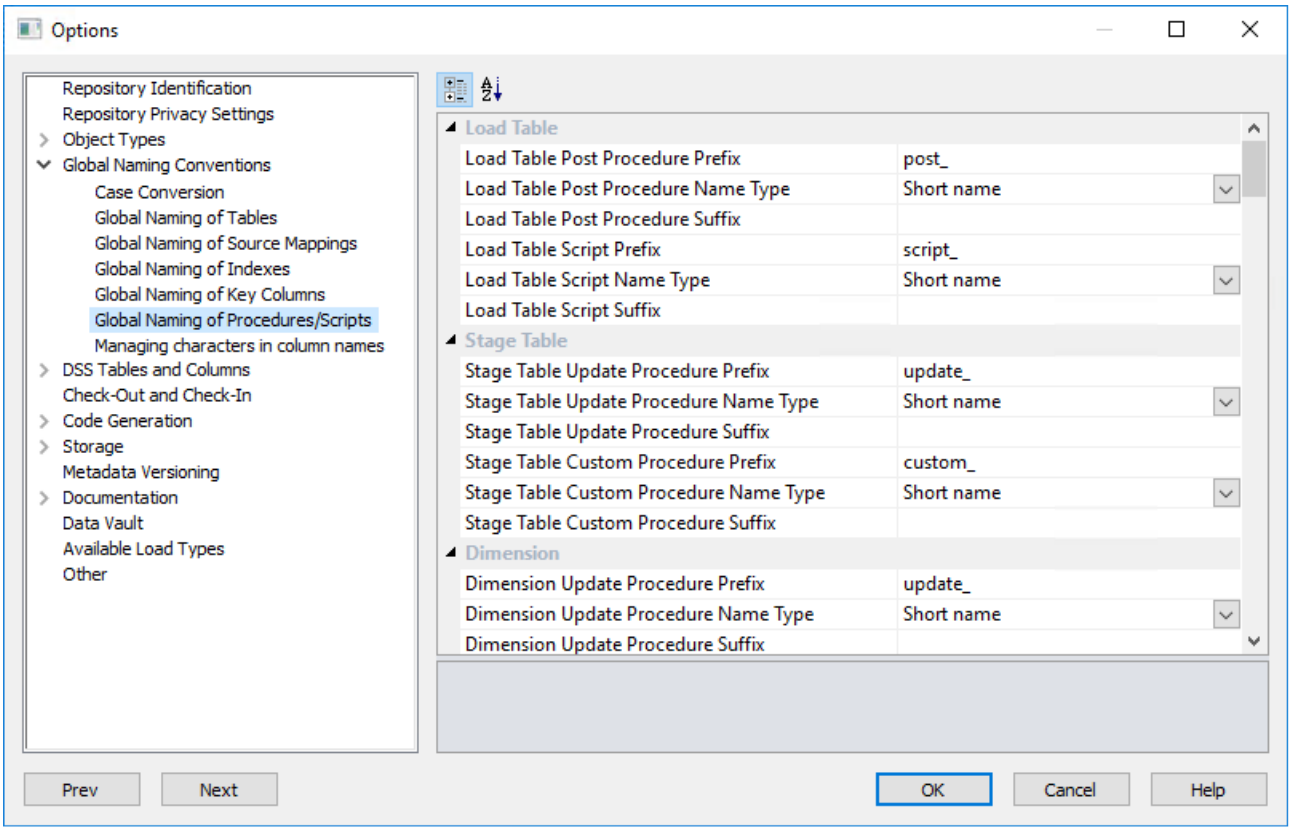
The default naming conventions for generated procedures can be set through the **Tools > Options** menu.

To generate a procedure, select the **(Build Procedure...)** option from the **Update Procedure** drop-down list in the table's **Properties** screen.

Procedure name defaults

The screen shown below is displayed in response to the procedure name defaults menu option. It provides a means of setting the naming defaults for all types of generated procedure.

The values as shown below are the default settings, but may be changed to meet the site requirements. The only restriction is on the size of the resultant name, which is database dependent.



The contents of the **prefix** and **suffix** fields must contain characters that are valid in a database stored procedure name and should preferably not contain spaces.

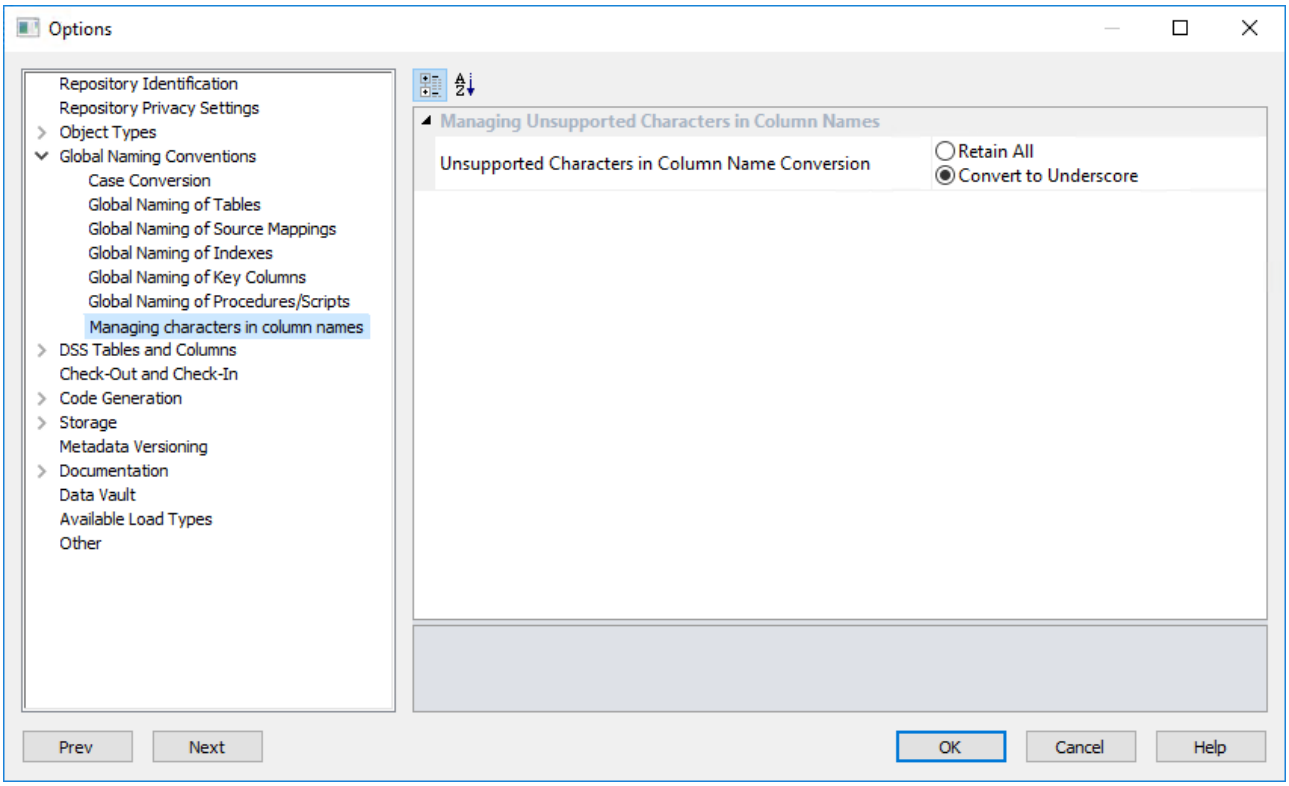
The **Name Type** may be either the full table name or the unique short name assigned to each table. In the case of smaller table names, the short name is usually the same as the table name.

For example, if we have a dimension called dim_product, then from the example screen above the three possible generated procedures would be called update_dim_product, custom_dim_product and get_dim_product_key.

© Copyright, WhereScape Software Ltd All rights reserved.

Managing Characters in Column Names

This option enables you to set the default action for **Managing characters in column names**.



This option enables you manage unsupported characters in column names when loading data from a source table, e.g. spaces in the source column names.

Option	Description
Retain All	Set this field if you want to retain all characters in the column name conversion, when you create a **Load** table from a source table with unsupported characters in its column names.
Convert to Underscore	This is the default setting. RED replaces the spaces in the source column names, when you create a **Load** table via drag and drop of a source table from the right pane to the middle pane of the Builder window.

DSS Tables and Columns

When building the data warehouse, WhereScape RED makes use of a number of special tables and columns. The two tables used are called by default **dss_source_system** and **dss_fact_table**. These tables are discussed in detail in the sections below.

The special columns used are defined in the table below.

Column name	Description
dss_batch	Not used at this stage.
dss_source_system_key	Added to support dimensions that cannot be fully conformed, and the inclusion of subsequent source systems. See the section below for more details.
dss_fact_table_key	Used in composite rollup fact tables to identify the source fact table that contributed the particular row.
dss_create_time	Indicates when a record was created.
dss_update_time	Indicates when the record was last updated in the data warehouse. Used in the

© Copyright, WhereScape Software Ltd All rights reserved.

Column name	Description
	update of rollup fact tables and aggregate tables.
dss_count	Applied to fact tables. Provides a simple row count variable that can be used by end user tools.
dss_current_flag	Used for slowly changing dimensions. This flag identifies the current record where multiple versions exist.
dss_version	Used for slowly changing dimensions. This column contains the version number of a dimension record. Numbered from 1 upwards with the highest number being the latest or current version. It forms part of the unique constraint for the business key of a slowly changing dimension.
dss_start_date	Used for slowly changing dimensions. This column provides a date time stamp when the dimension record began life. If null then it was the first record. It is used to ascertain which dimension record should be used when multiple are available.
dss_end_date	Used for slowly changing dimensions. This column provides a date time stamp when the dimension record ceased to be the current record. It is used to ascertain which dimension record should be used when multiple are available.
dss_change_hash	Used for a Satellite table. This column identifies the differences in the descriptive columns of a Satellite table which is used for generating the change hash key for creating a Satellite object.

All of these special columns and tables can be renamed through the **Tools > Options > DSS Tables and Columns** menu option. The columns can simply be renamed, but the tables require valid table names that meet certain criteria. Refer to the related sections below.

Note:

When using table names other than the defaults for dss_source_system and dss_fact_table it is worth considering the fact that by default, the metadata backups will include any table that begins with "dss_". Therefore, if a table is used it is recommended that it have a name starting with "dss_". The advantage is that a working meta repository will be established through a backup and restore, if these tables are included in the backup set.

© Copyright, WhereScape Software Ltd All rights reserved.

dss_source_system

This pseudo dimension is designed to identify a data source for a dimension row. Its purpose is to handle non conformed dimensions, or changes in source systems. If its use is not desired (default), then leave this field blank.

For example:

An organization has a number of factories. These factories are referenced by all of the operational systems. The production system has its own code for each factory, and this is the unique means of identifying the factory. The distribution system has a factory short name which it uses for the unique identifier. The raw materials system simply uses the factory name. It is probably not practical or even desirable to force these source systems to utilize a standard factory identification method, so instead we allow the dimension to be non-conformed. We do however, insist on a standard factory name convention, so that our reports and queries will join information when the factory name is used.

In this example, the dss_source_system_key is used to identify the source of the data for the dimension row. It also adds to the unique business key, so that two source systems can utilize the same code to refer to different entities. This key also provides a degree of future proofing in the data warehouse, to assist in the possible changing of an underlying source system.

The generated procedure code always set the key value of this table to 1. Therefore, manual code changes is required to make use of the functionality that this table offers.

When dimensions are built, the dss_source_system_key is added by default. It can be deleted if the need for such a feature is not seen.

If this table is to be given a different name ,then it and all its columns can be renamed, or the following steps can be taken:

1. Create a new table by dragging the column dss_source_system_name from dss_source_system into a dimension target.
2. Change the object type from dimension view to dimension and specify the new table name. (see note above on the use of "dss_").
3. Rename the dss_source_system_name column to match the new table name.
4. Delete the last two columns.
5. Under the table Properties, change the table type to **Mapping table**. This prevents the table from being seen as a dimension in the documentation.
6. Change the dss_source_system table name in the screen above, via the **Tools>Options>DSS Tables and Columns** defaults menu option.

dss_fact_table

This pseudo dimension is used internally by the WhereScape RED generated procedures to assist in the updating of rollup fact tables. It provides a means of identifying which fact table contributed a particular row to a composite rollup fact table. See the section on rollup fact tables for an explanation of how the generated procedures operate.

This table can be renamed, or a new table created. The table identified as the 'dss_fact_table' must, however, conform to a number of standards if the generated code is to compile and work correctly.

The following columns must exist, where table_name is the new name of the table:

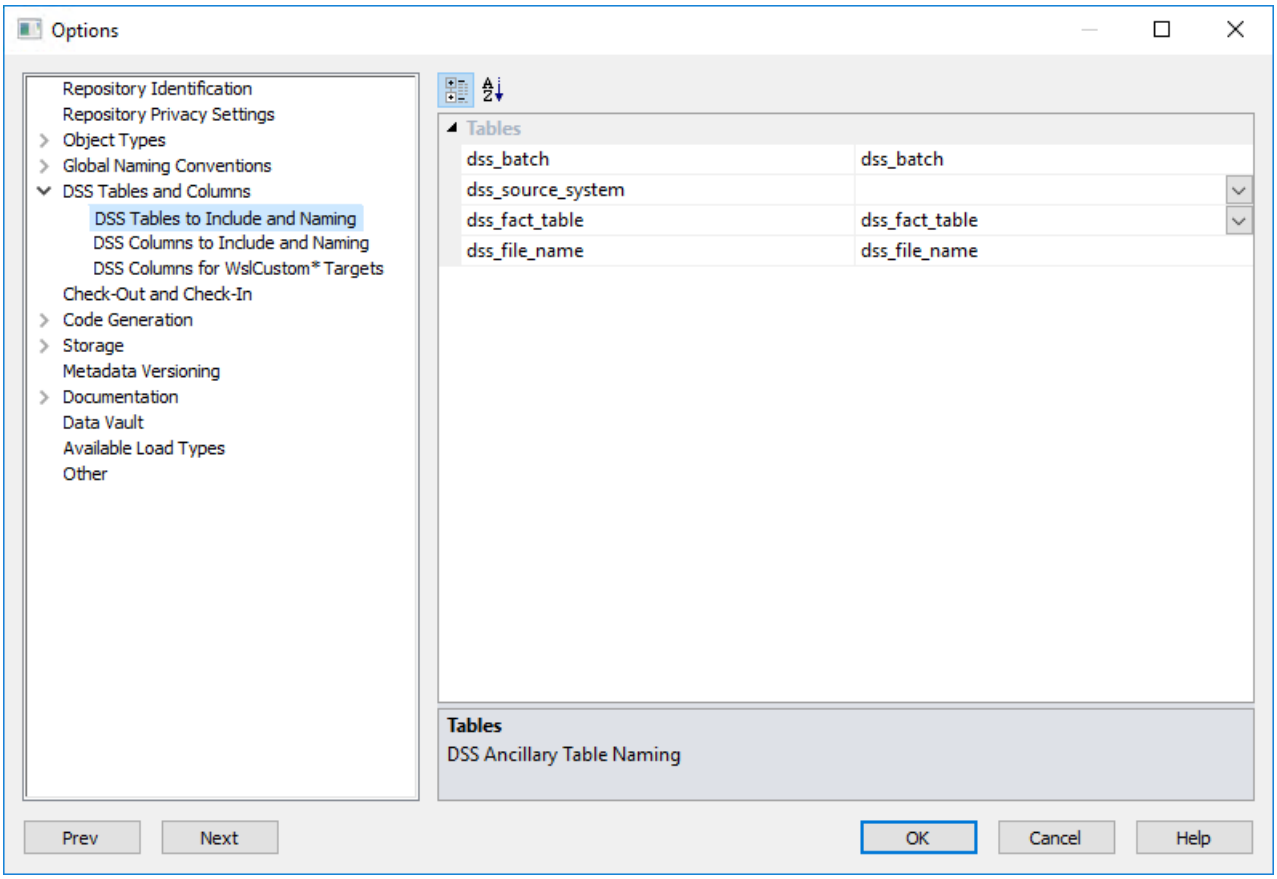
Column name	Purpose
table_name_key	The artificial key for the table. Must be a numeric data type.
table_name_name	The name of the fact table. Must be at least a varchar2(64) data type.
table_name_type	The type of fact table. Must be at least a varchar2(24).

The generated procedures automatically add rows to this table, as required. As with the dss_source_system table, the table type for this table must be set to **Mapping table** to prevent it from being seen as a Dimension in the documentation.

© Copyright, WhereScape Software Ltd All rights reserved.

DSS Tables

This option enables you to set the **DSS Tables**.



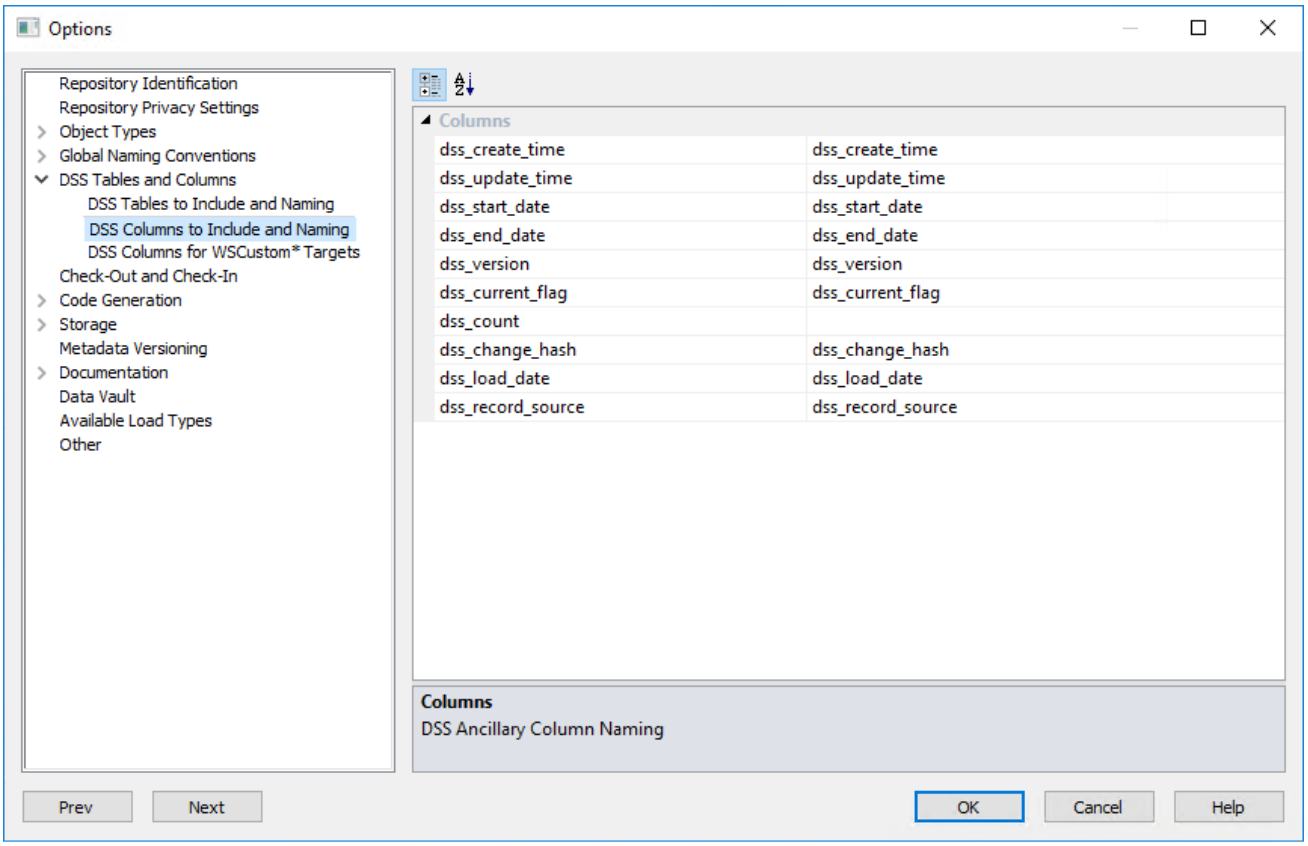
© Copyright, WhereScape Software Ltd All rights reserved.

Tables

Set the **DSS Tables**.

DSS Columns

This option enables you to set the **DSS Columns**.



Columns

Set the **DSS Columns**.

Fields	Description
dss_create_time	Column added to all Stage, ODS, EDW 3NF, Dimension, Fact and Aggregate tables for information purposes only. Leave the field blank to deactivate or add a name for the dss_create_time column, default is dss_create_time.
dss_update_time	Column added to all dimension, stage and fact tables. It is required if the generated code for fact and aggregate tables is to be used.
dss_start_date	Column used for slowly changing dimensions. It is used to identify when a dimension row was replaced. This is a required field.
dss_end_date	Column used for slowly changing dimensions. It is used to identify when a dimension row was replaced. This is a required field.
dss_version	Column used for slowly changing dimensions. Is it used to store the version of a dimension row. This is required for unique constraints.
dss_current_flag	Column used for slowly changing dimensions. It is used to identify the current dimension row. This is a required field.
dss_count	Column added to fact tables to provide a simple row counter. This is required but can be deleted after generation. Leave the field blank to deactivate.
dss_change_hash	Column used to identify the differences in the descriptive columns of a Satellite table which is used for generating the change hash key for a Satellite object. Refer to **Creating Data Vault Stage Tables** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

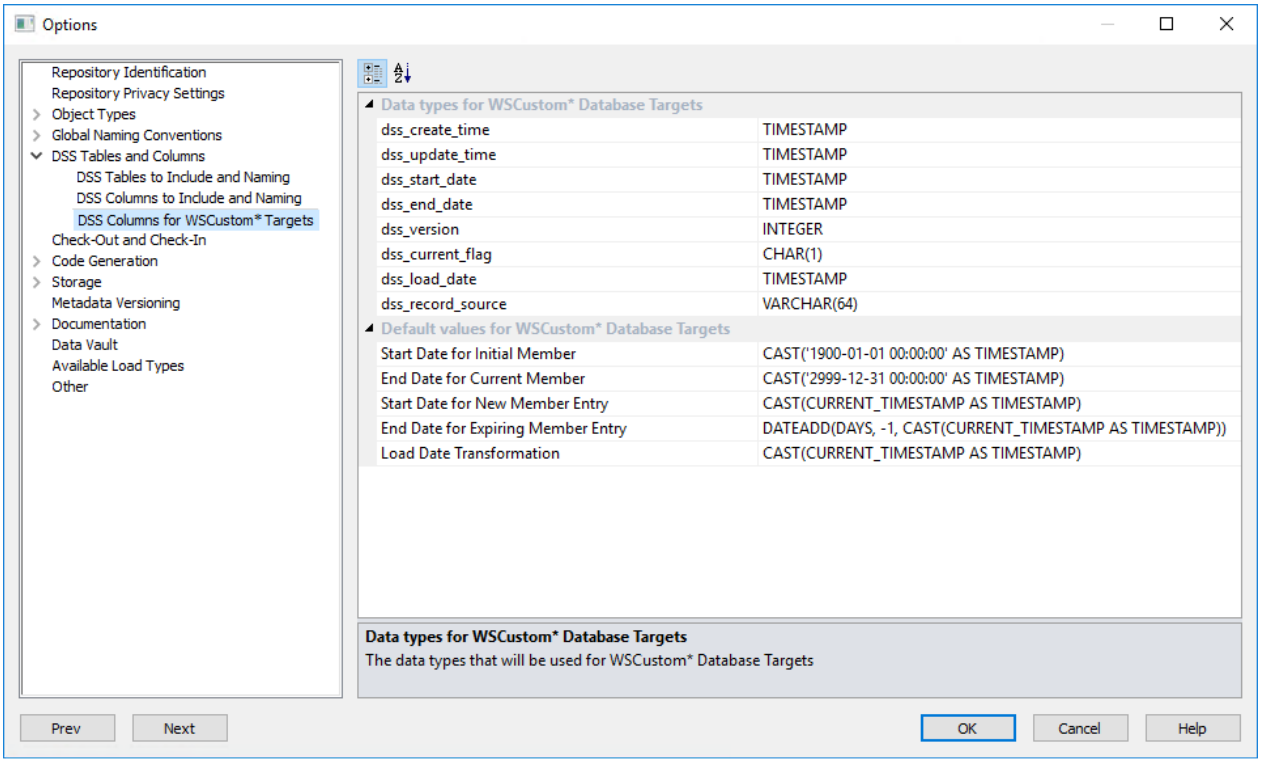
Fields	Description
dss_load_date	Column used to store the date when the value in this row was loaded into the metadata repository. This column and the **dss_record_source** column below is added to new Load tables that have the option **Add meta data columns to table** selected. Refer to **Data Vaults** for details.
dss_record_source	Column used to store a descriptive term to identify the source of this record. This column and the **dss_load_date** column above is added to new load tables that have the option **Add meta data columns to table** selected. Refer to **Data Vaults** for details.

DSS Columns for Custom Targets

Note:

These settings are only available/displayed if your WhereScape RED license includes support for custom target databases.

This option enables you to set the **DSS Columns** for custom database targets.



Data types for Custom Database Targets

Enables you to set the data types that are used in the DSS columns added to new table objects.

Notes:

1. The **Hash Key Generation Wizard** creates new hub and link hash keys in a Data Vault Stage table with the same data type defined for the **dss_change_hash** column, if the table is in a Custom database target.
2. If the table is **not** in a Custom database target, the data type CHAR(32) is used.
3. Changing this data type has no impact on any existing hash key columns.

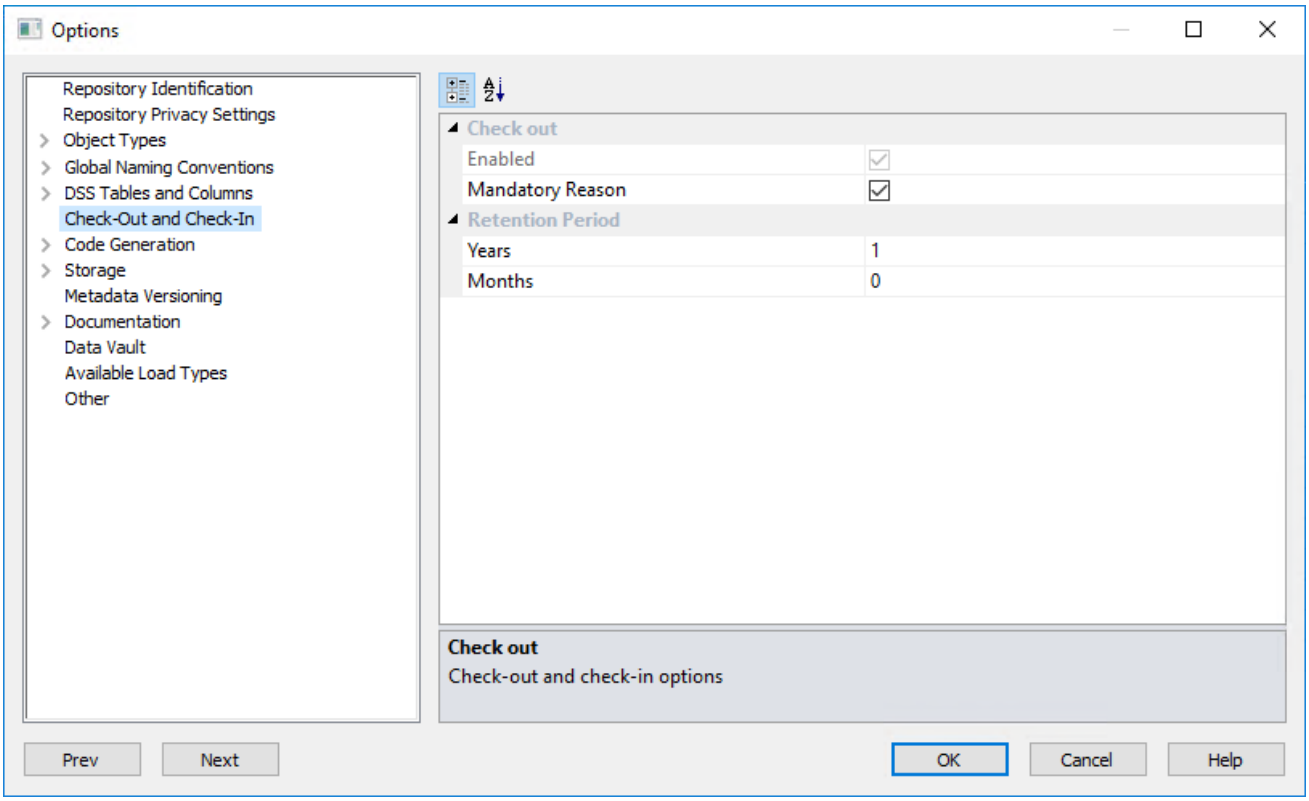
Fields	Description

© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
Default values for Custom Database Targets	Enables you to set the default values that are used when generating update procedures for Slowly Changing Dimension tables and Load tables.
Start Date for Initial Member	Default date function to use for the start date of rows added in a Slowly Changing Dimension table.
End Date for Current Member	Default date function to use for the end date of current rows in a Slowly Changing Dimension table.
Start Date for New Member Entry	Default date function to use for the start date of current rows in a Slowly Changing Dimension table.
End Date for Expiring Member Entry	Default date function to use for the end date of expiring rows in a Slowly Changing Dimension table. **Note** Refer to the **Change Detection tab** section under **Generating the Dimension Update Procedure** for details.
Load Date Transformation	Default date function to use for populating the new records inserted in Load tables.

Check-Out and Check-In

This option enables you to set up for the **Checking-out** or **Checking-In** of Procedures.



Check out

- **Enabled**: This option is selected by default to enable procedures to be checked-in or checked-out.
- **Mandatory Reason**: Select this option if a reason is mandatory for checking out or checking in procedures, else deselect.

Retention Period

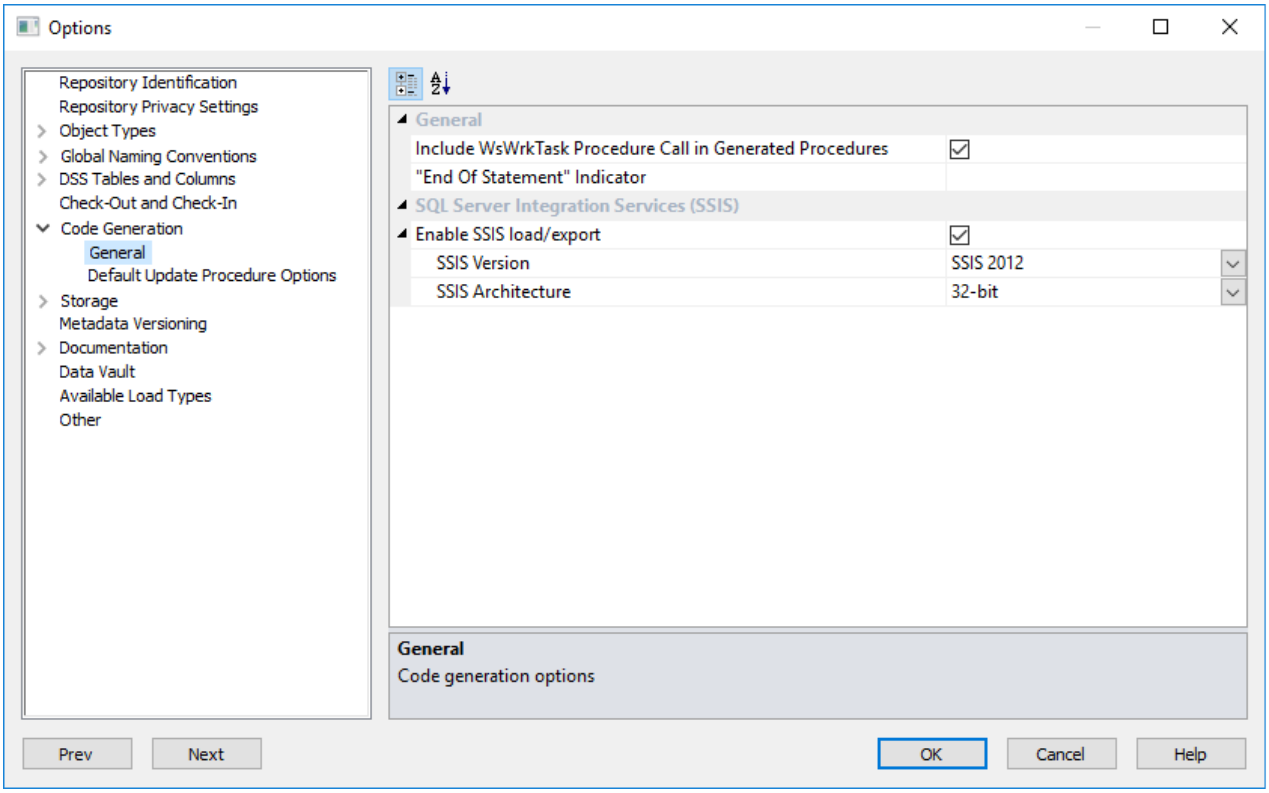
Set the length of time; **Years** and **Months**, for which procedures may be checked-out.

© Copyright, WhereScape Software Ltd All rights reserved.

Code Generation

General

This option enables you to set some general **Code Generation** settings.



General

- **Include WsWrkTask Procedure**: This option is selected by default and results in a call to the WsWrkTask function being placed at the end of most of the generated update procedures. These calls to WsWrkTask result in counters being set in the meta table ws_wrk_task_log. These counters can be viewed via a query on the view **ws_admin_v_task**.
- **"End of Statement" Indicator**: Set the indicator to separate multiple SQL statements in a SQL block. If left blank, the default value of <EOS> is used.

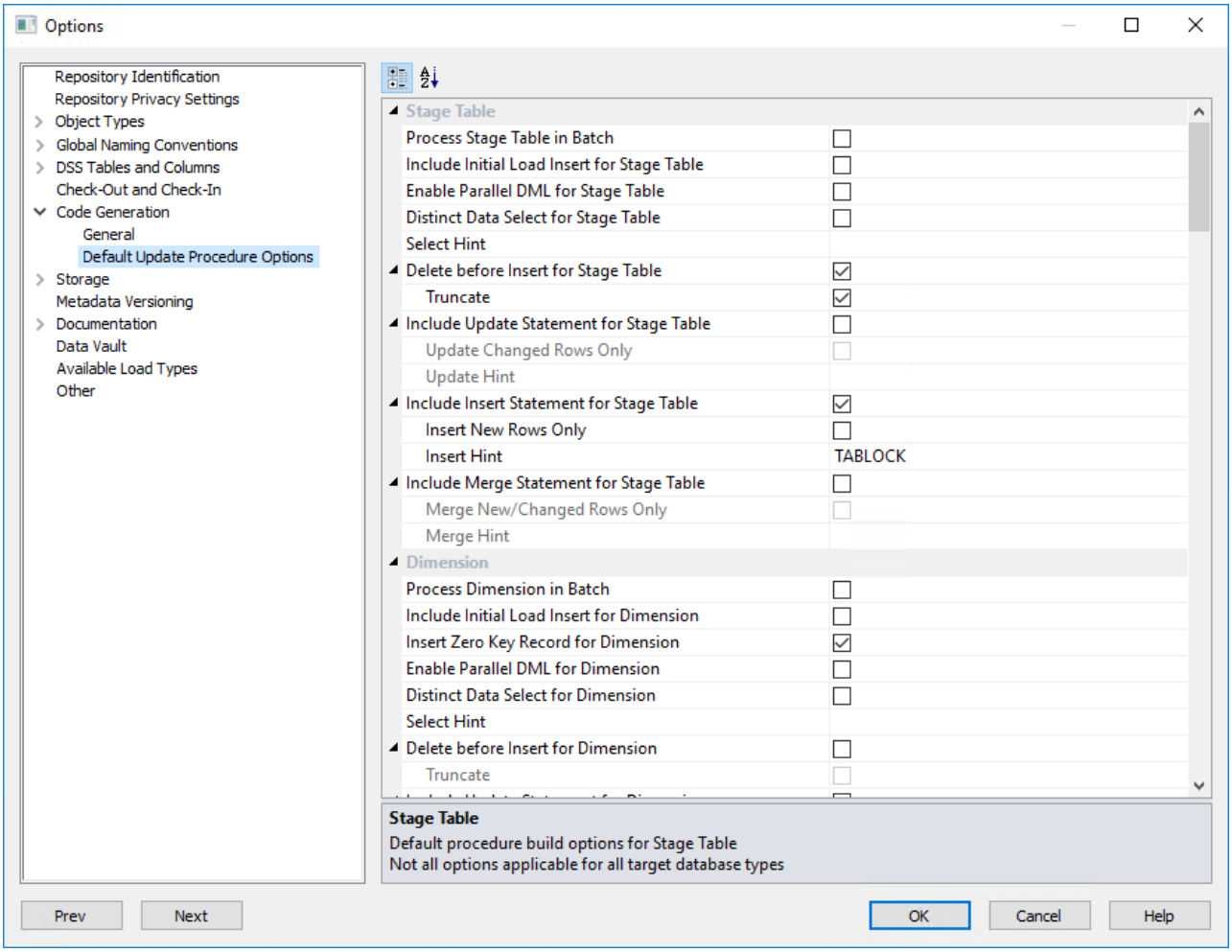
SQL Server Integration Services

If the **Enable SSIS load/export** option is selected, you must specify the **Version** and **Architecture** of the SSIS being used. To use SSIS to load or export data, the relevant version and architecture of the SSIS must be selected from the corresponding drop-down lists.

© Copyright, WhereScape Software Ltd All rights reserved.

Default Update Procedure Options

This option enables you to set some **default update procedure** settings for specific object types. Not all options are relevant for all combinations of database, object type and build type.
The screen below shows an example of an **Oracle** repository's **Default Update Procedure Options** screen.



Default Update Procedure Options

Fields	Description
Process in Batch	Enables users to select a column to drive data processing in a loop based on the distinct ordered values of the selected column.
Include Initial Load Insert	Enables users to add an additional insert statement to the update procedure. If the target table is empty, the new insert statement is run in place of the standard generated code.
Insert Zero KeyRecord	Enables users to add an insert statement for an unknown record with an artificial key of zero. Only applicable to tables with an artificial key. Oracle: **Enable Parallel DML** - enables users to add all code required to the update procedure for enabling Oracle parallel inserts. This option is only available for **Oracle**.
Distinct Data Select	Enables users to ensure duplicate rows are not added to the table.
Select Hint	Enables users to enter a database-compliant hint to be used in the SELECT statement. Parameters $TABLE$ and $INDEXS is automatically replaced at procedure generation time.
Delete before Insert	Enables users to delete statement to be added to the update procedure before any update or insert statement.

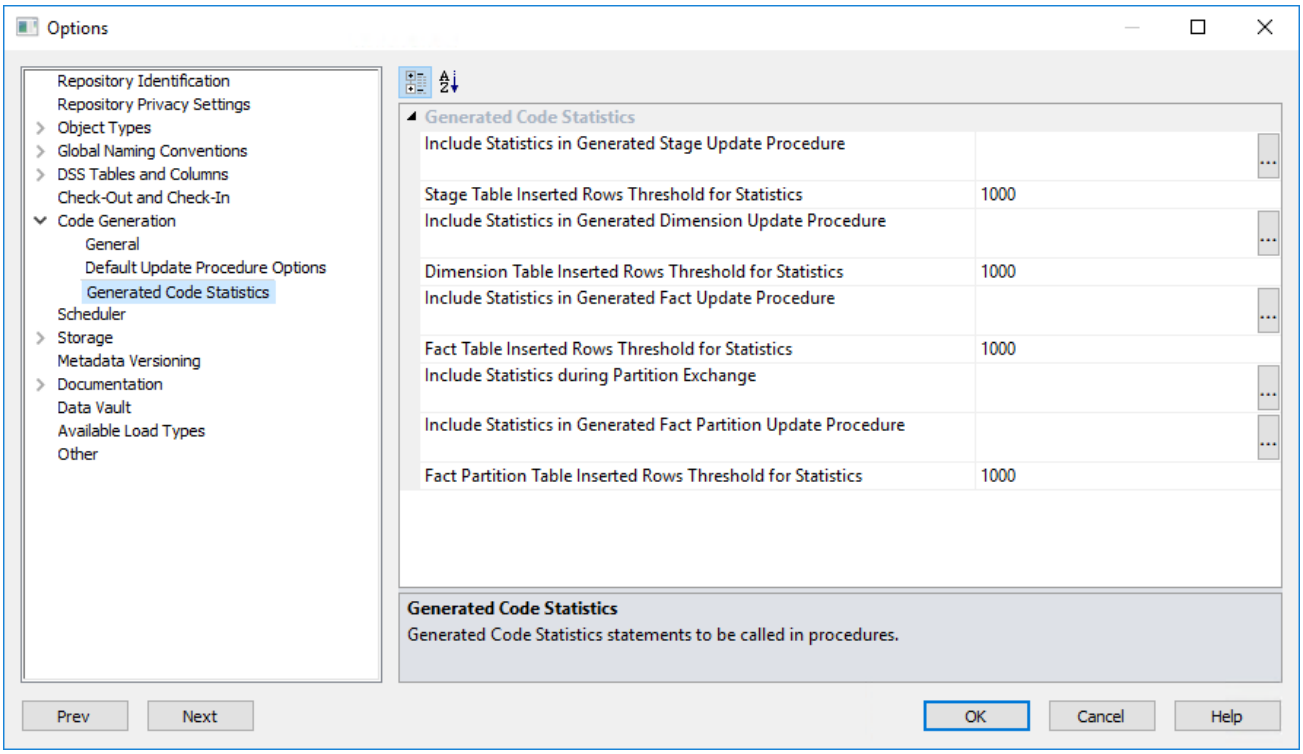
© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
Truncate	Performs a DDL operation to delete all records from a table, rather than a predicate based DML operation to delete individual rows. This option is automatically enabled when **Delete before Insert** is enabled, but can be deactivated separately.
Include Update Statement	Enables users to include an update statement in the procedure to update changing rows in the table.
Update Changed Rows Only	Enables users to use change detection to work out what rows require updating.
Update Hint	Enables users to enter a default database hint to be used in the UPDATE statement. This option is available for **SQL Server** and **Oracle** only. Oracle default hint is APPEND. SQL Server default hint is TABLOCK.
Include Insert Statement	Enables users to include an insert statement in the procedure to insert new rows in the table.
Insert New Rows Only	Enables users to use change detection to work out what rows require inserting.
Insert Hint	Enables users to enter a default database hint to be used in the INSERT statement. This option is available for **SQL Server** and **Oracle** only. Oracle default hint is APPEND. SQL Server default hint is TABLOCK.
Include Merge Statement	Enables users to include a merge statement in the procedure to merge new/changed rows in the table.
Merge New/Changed Rows Only	Enables users to use change detection to interpret which rows require merging.
Merge Hint	Enables users to enter a default database hint to be used in the MERGE statement. This option is available for **SQL Server** and **Oracle** only. Oracle default hint is APPEND. SQL Server default hint is TABLOCK.

© Copyright, WhereScape Software Ltd All rights reserved.

Generated Code Statistics

This option is only available for **Oracle** databases.



Generated Code Statistics

Fields	Description
Include Statistics in Generated Stage Update Procedure	Sets the statistics command to be included in the automatically generated procedure for Stage tables. Parameter $TABLE$ is automatically replaced by the table name. The key word $UNUSED$ removes the statistics portion from the generated code in the procedure.
Stage Table Inserted Rows Threshold for Statistics	Sets the threshold parameter to be used by the four statistic commands that are inserted into the automatically generated procedures. It indicates how many rows are to be inserted before the statistic commands are executed.
Include Statistics in Generated Dimension Update Procedure	Sets the statistics command to be included in the automatically generated procedure for Dimension tables. Parameter $TABLE$ is automatically replaced by the table name. The key word $UNUSED$ removes the statistics portion from the generated code in the procedure.
Dimension Table Inserted Rows Threshold for Statistics	Sets the threshold parameter used by the four statistic commands that are inserted into the automatically generated procedures. It indicates how many rows are to be inserted before the statistic commands are executed.
Include Statistics in Generated Fact Update Procedure	Sets the statistics command to be included in the automatically generated procedure for Fact tables. Parameter $TABLE$ is automatically replaced by the table name. The key word $UNUSED$ removes the statistics portion from the generated code in the procedure.
Fact Table Inserted Rows Threshold for Statistics	Sets the threshold parameter to be used by the four statistic commands that are inserted into the automatically generated procedures. It indicates how many rows are to be inserted before the statistic commands are executed.
Include Statistics during Partition Exchange	Sets the statistics command to be included in the automatically generated procedure during partition exchanges. Parameter $TABLE$ is automatically replaced by the table name. The key word $UNUSED$ removes the statistics portion from the generated code in the procedure.

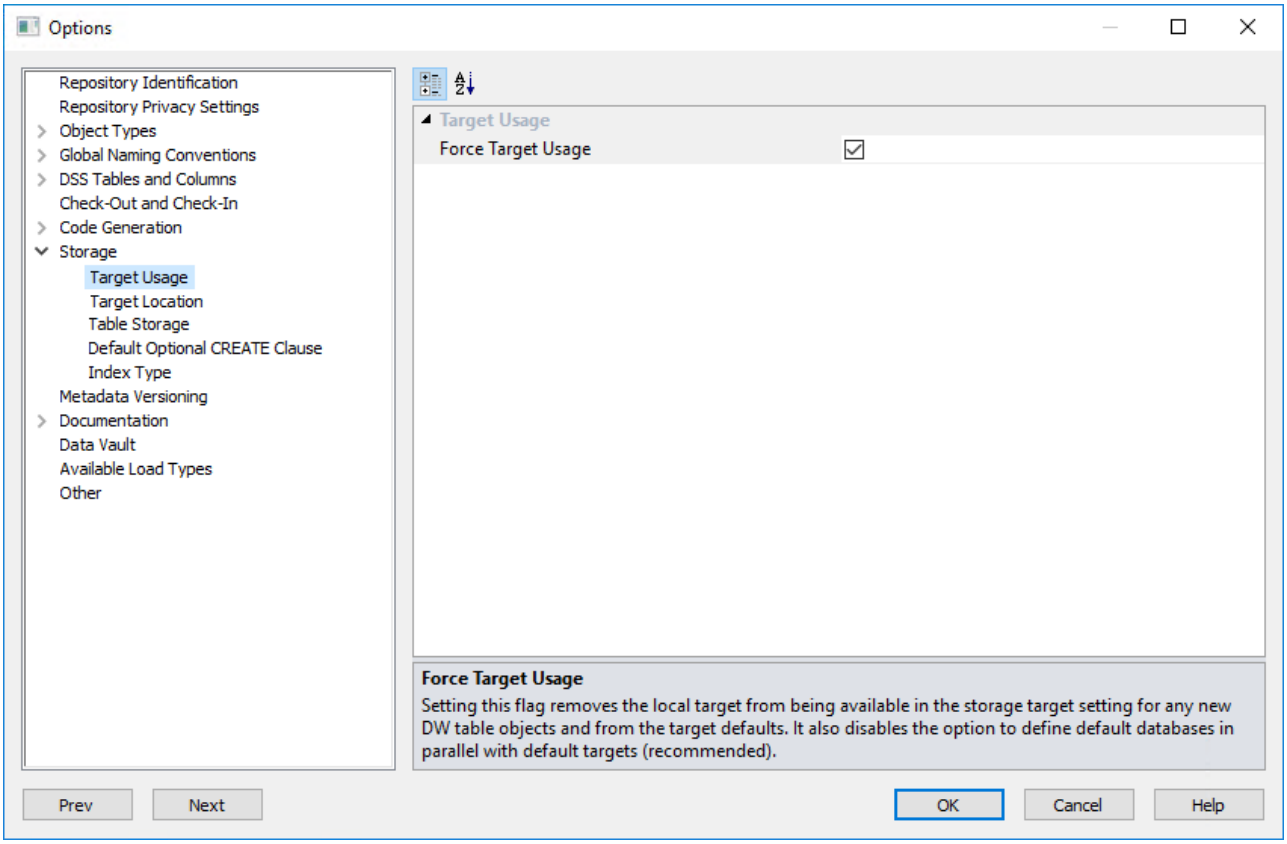
© Copyright, WhereScape Software Ltd All rights reserved.

Storage

Target Usage

This option enables you to force the use of table target locations (defined in the **Connection Properties > Target Settings**), and prevents the use of legacy local targets when creating new table objects in RED.

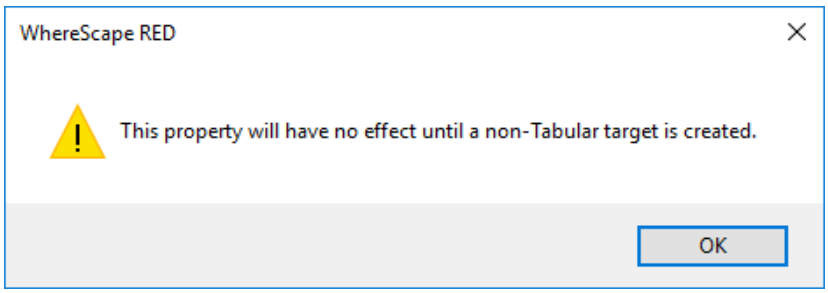
This option is enabled by default for new repositories—it is recommended to define targets for the data warehouse objects that is separate from the metadata database.



The **Force Target Usage** setting removes the option to use local targets in the **Default Target** drop-down list of the **Target Location** option and also hides the **Table Storage** option, which is used to set the storage locations for each table object type created in RED.

Notes:

- Existing RED objects or objects loaded into RED with **local** set as their target, retains the local target setting.
- If the target location is changed from local to a specific target, then there is no option to set it back to local without deactivating the **Force Target Usage** setting.
- If table target(s) has not been defined, a message is displayed to warn users that this setting does not take effect, until a target is created.



To see more about creating target locations, Refer to **Connection to the Data Warehouse** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

Target Location

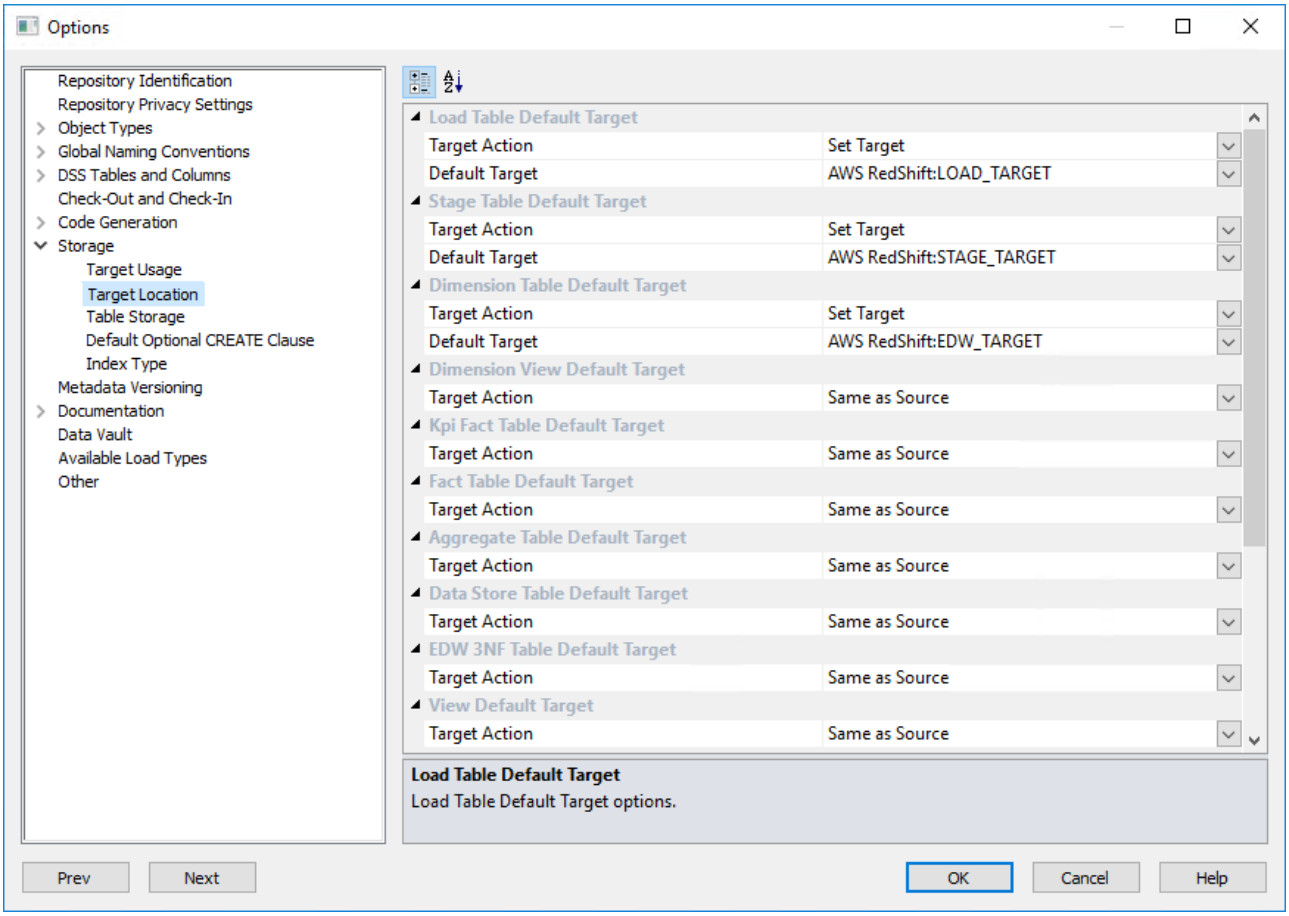
Target Location options enable users who are placing objects across multiple schemas to set default target locations for new tables.

Note:

Refer to **Distributing Tables across Multiple SQL Server Databases** if using multiple databases as target locations for tables in an SQL environment.

Default table target locations can be set for the following objects:

- Load
- Stage
- Dimension
- Dimension View
- KPI Fact
- Fact
- Aggregate
- Data Store
- EDW 3NF
- View
- Hub Table
- Satellite
- Link
- Custom



© Copyright, WhereScape Software Ltd All rights reserved.

Target Action

Set Target

This option enables you to set a default target for new tables to be created. It enables the **Default Target** drop-down list where a specific target for new tables can be defined.

Note:

If the **Force Target Usage** option is enabled in the **Target Usage** setting, selecting **Set Target** automatically sets the **Default Target** field with the first target location value, available from the drop-down list. The option to use **Local** target is also not available.

Same as Source

This option is selected if the table's default storage must be the same as the original source where the table is coming from. This option cannot be selected for Load tables.

Default Target

A default target can only be entered if the **Set Target** action has been selected in the **Target Action** drop-down list.

With this option, users can choose between setting a table's default location:

- (**local**) if the **Force Target Usage** setting is not enabled in the **Target Usage** setting, or
- to any other **target locations** that have been defined in the relevant connection Properties.

To set a default target location on a table by table basis:

1. Select the **Set Target > Same as Source** option from the **Target Action** drop-down list.
2. To have tables located in a specific target, select a **default target** where the new object is stored, when the object is dragged and dropped to the middle work pane.

To know more about Target table locations, refer to **Connections to the Data Warehouse/Metadata** Repository for SQL and Oracle Databases and **Connections - Database** or **Connections - ODBC** for target based databases, such as Netezza, Greenplum and PDW.

Even though the default target location can be set in the **Target Location** options, this setting can also be changed after the table has been created via the **Storage** tab of each table object's **Properties** screen.

To see more information about changing the schema after a table has been created, refer to **Storage**.

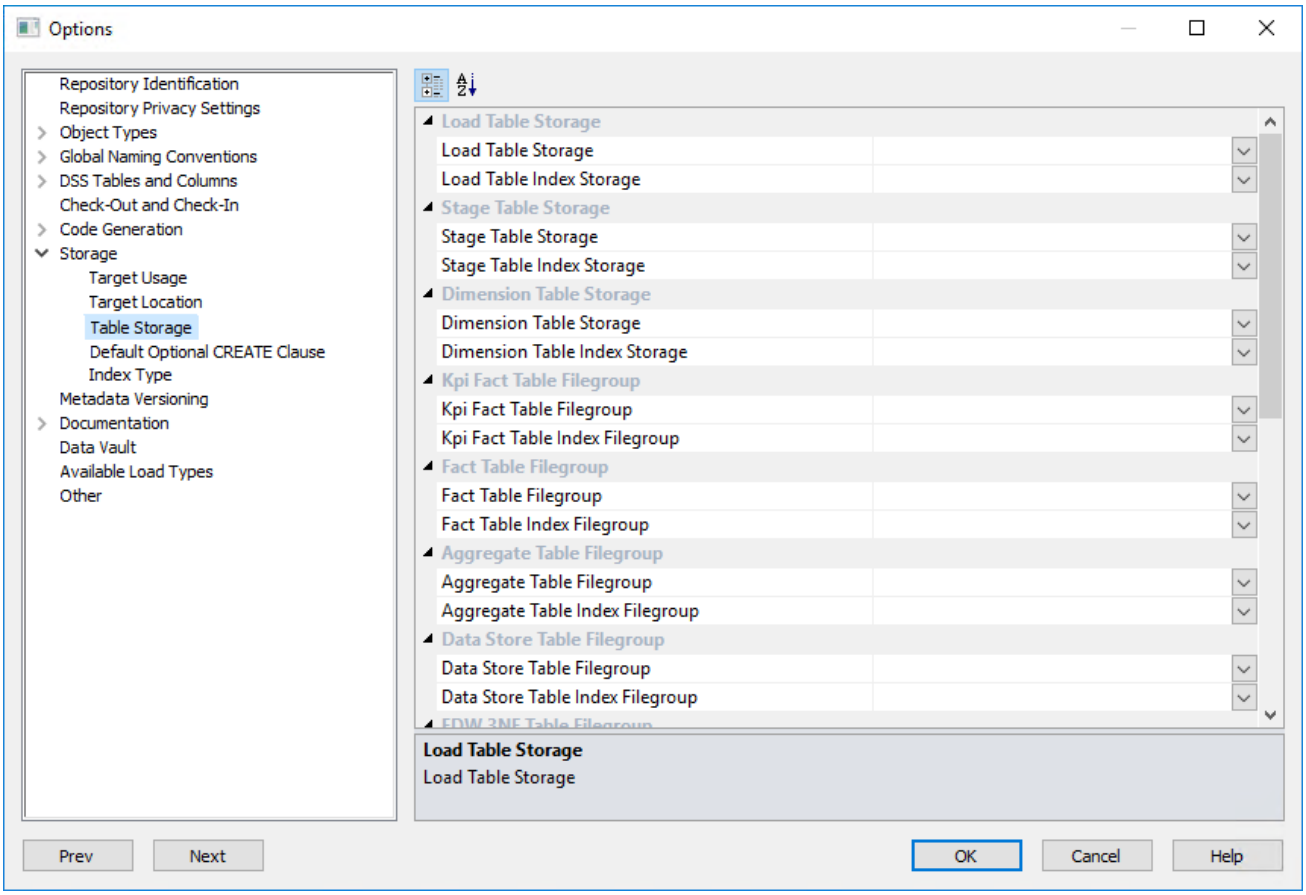
© Copyright, WhereScape Software Ltd All rights reserved.

Table Storage

Note

The **Table Storage** option is only available/visible if the **Target Usage > Force Target Usage** setting is not enabled.

This option enables you to set the **Storage** locations for each table object type created in RED.



Set the **Storage** locations for each table type.

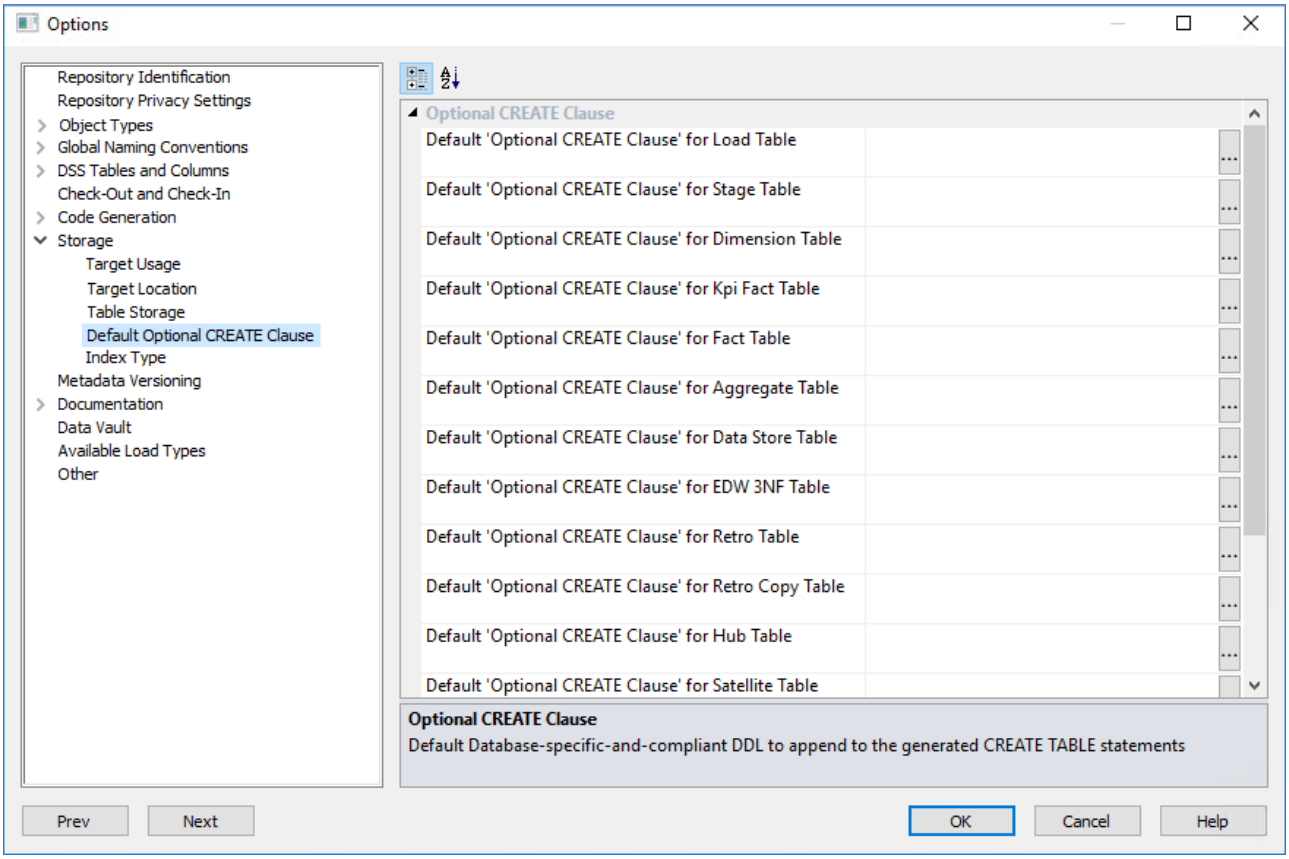
These defaults are applied when a table is created. They can be changed by selecting the **Storage** tab on the **Properties** screen of a table.

© Copyright, WhereScape Software Ltd All rights reserved.

Default Optional CREATE Clause

This option enables you to define a default value for the "Optional CREATE Clause" property of each object type, which is populated when the object is first created.

The Optional CREATE Clause text is appended to the DDL CREATE statement when the table is generated. An example use of this option is to define default settings for parallelism and logging in an Oracle database—see below for further details.



Tip:

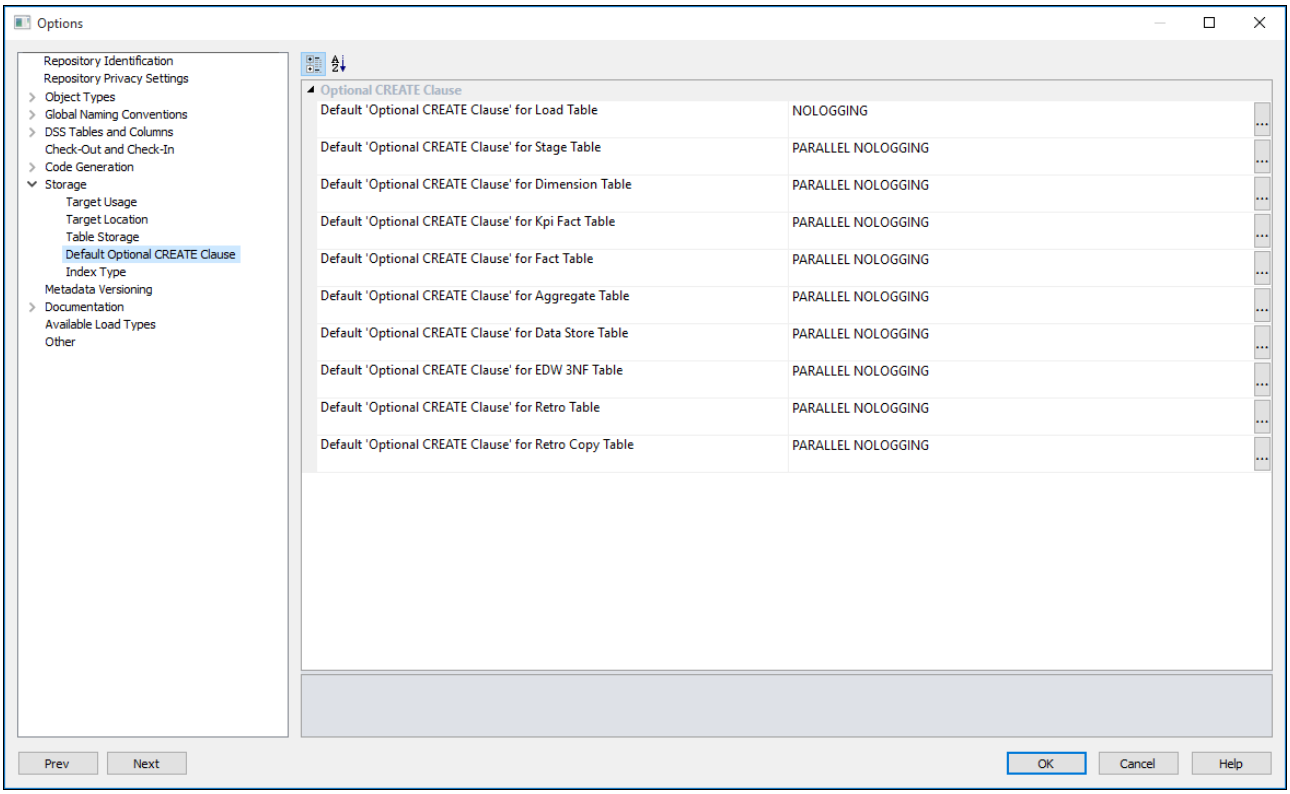
This option is only used to set the default optional create clause for new objects.
To edit the Optional CREATE Clause of an existing object or edit the clause on a table by table basis, go to the object's Properties screen, click the Storage tab and edit the Optional CREATE Clause field.

© Copyright, WhereScape Software Ltd All rights reserved.

ORACLE

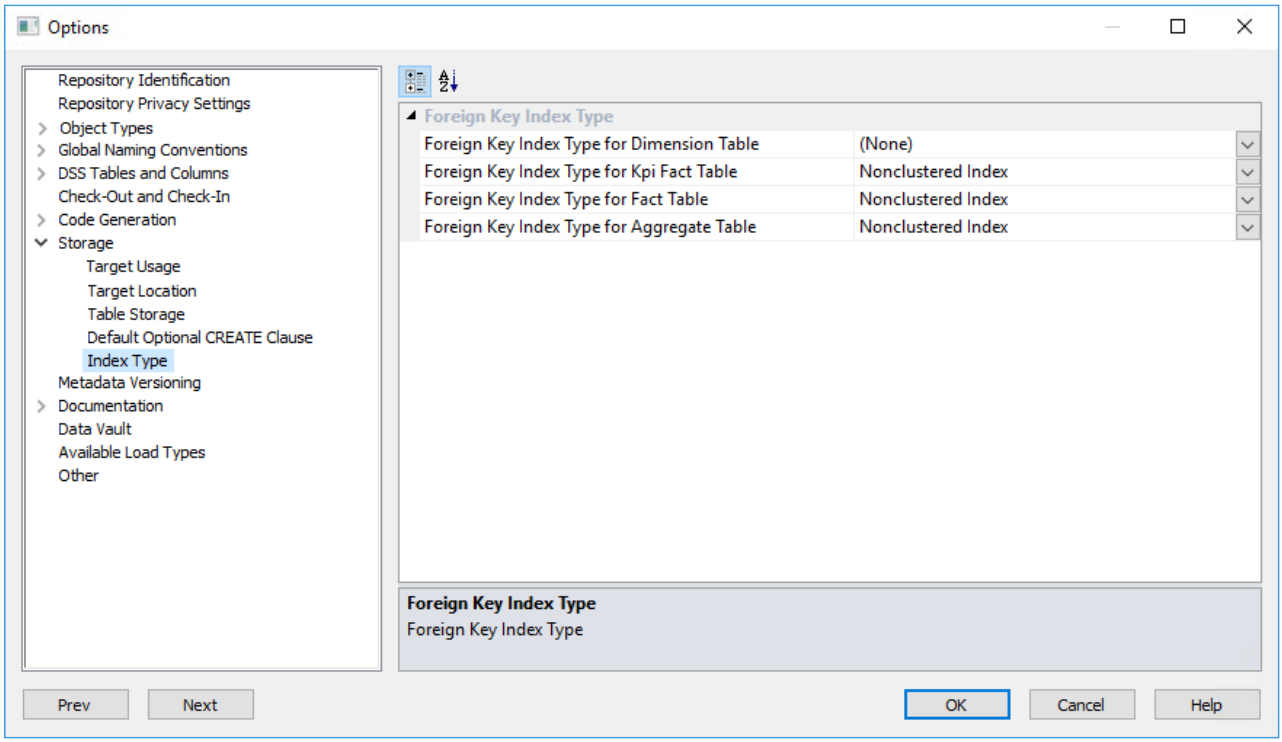
Oracle's default values are set to NOLOGGING for Load Tables and PARALLEL NOLOGGING for remaining objects.

To edit the default optional create clause for each object, replace the statement on the free text field or click on the ellipsis button.



Index Type

This option enables you to set the default type of **Foreign Key Index Type** for each table type.



© Copyright, WhereScape Software Ltd All rights reserved.

Foreign Key Index Type

Set the default index types built for foreign key columns of Dimension tables, KPI Fact tables, Fact tables and Aggregate tables.

SQL Serveroptions:

- (None) - will not be defined
- BTREE - this is a standard index per key column
- COVERING - this is a single index with all the key columns in it

Oracle options:

- (None) - will not be defined
- BITMAP - the standard Oracle bitmap index
- BTREE - this is a standard index per key column

These defaults are applied when an index definition is created.

They can be changed by selecting the **Storage** tab in the **Properties** window of an index.

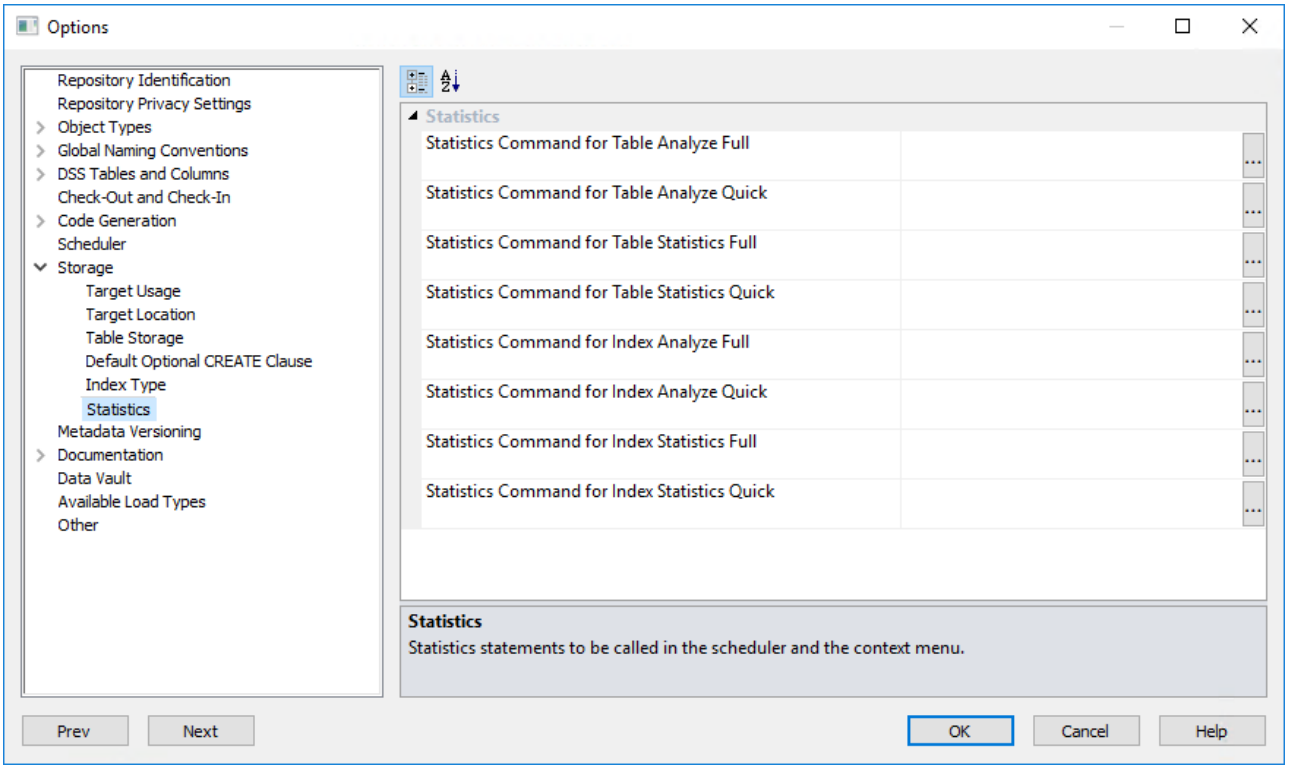
Statistics

The Scheduler Statistics option is only available for **Oracle** databases.

Statistics command entered here is executed when using the scheduler with Stats, Quick Stats, Analyze or Quick Analyze actions to gather statistics on tables.

Parameter **$TABLE$** is automatically replaced by the table name, parameter **$INDEX$** is replaced by the index name; and parameter **$SCHEMA$** is replaced by the owner name. Please see examples below.

For more information about adding statistics tasks to jobs, refer to **Editing Tasks in a Job f**or details.



© Copyright, WhereScape Software Ltd All rights reserved.

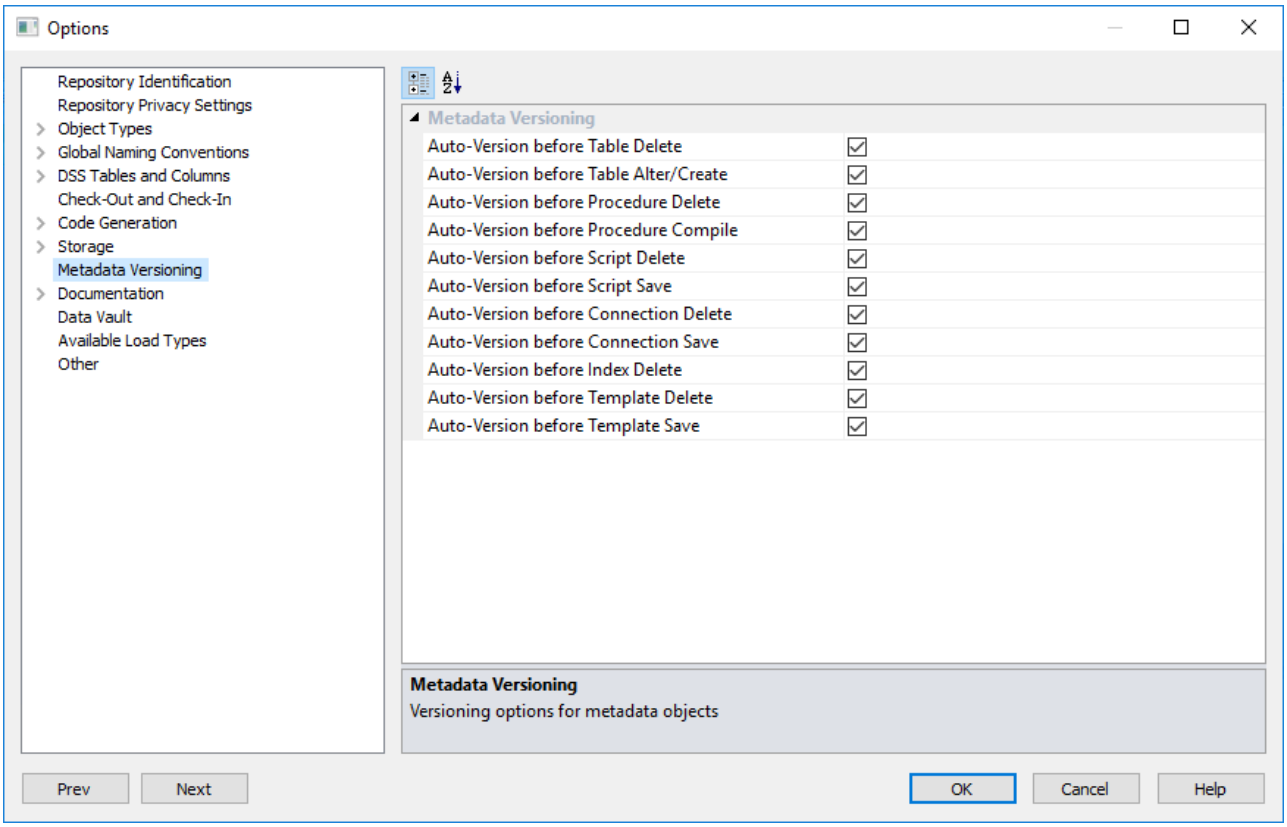
Examples:

Scheduler Statistics	
Statistics Command for Table Analyze Full	**analyze table $TABLE$ estimate statistics sample 35 percent**
Statistics Command for Table Analyze Quick	**analyze table $TABLE$ estimate statistics sample 1000 rows**
Statistics Command for Table Statistics Full	**BEGIN dbms_stats.gather_table_stats(ownname=>'$SCHEMA',tabname=>'$TABLE$',estimate_percent=>'50',degree=>16); END;**
Statistics Command for Table Statistics Quick	**BEGIN dbms_stats.gather_table_stats(ownname=>'$SCHEMA',tabname=>'$TABLE$',estimate_percent=>'10',degree=>16); END;**
Statistics Command for Index Analyze Full	**analyze index $INDEX$ estimate statistics sample 10 percent**
Statistics Command for Index Analyze Quick	**analyze index $INDEX$ estimate statistics sample 1 percent**
Statistics Command for Index Statistics Full	**BEGIN dbms_stats.gather_index_stats(ownname=>'$SCHEMA',indname=>'$INDEX$',estimate_percent=>'20',degree=>16); END;**
Statistics Command for Index Statistics Quick	**BEGIN dbms_stats.gather_index_stats(ownname=>'$SCHEMA',indname=>'$INDEX$',estimate_percent=>'5',degree=>16); END;**

Metadata Versioning

This option enables you to alter the **Metadata Versioning** settings.

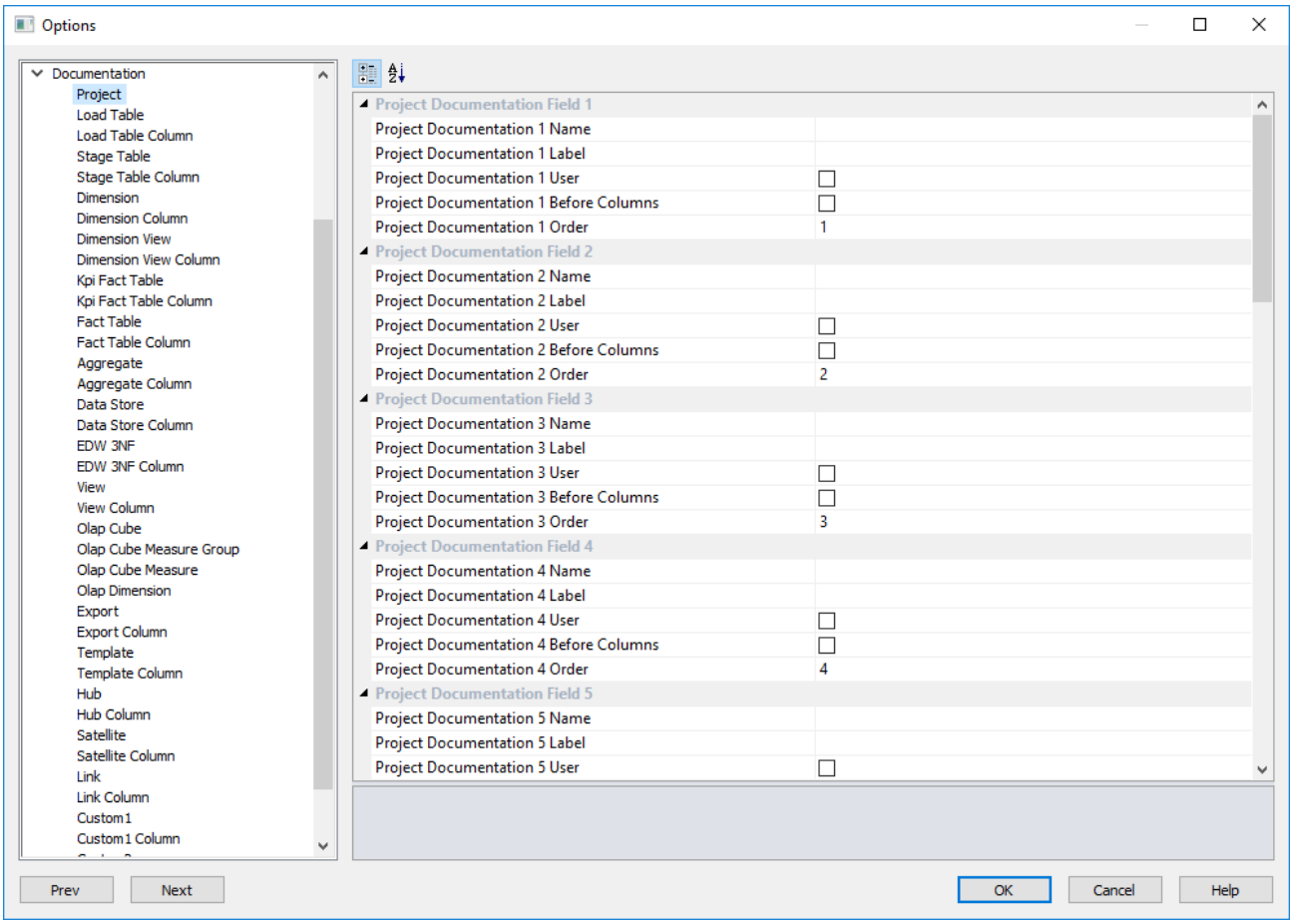
Select or clear the corresponding check box to set when to auto-version the metadata. All options are selected by default.



© Copyright, WhereScape Software Ltd All rights reserved.

Documentation

This options enables you to alter the **Documentation** settings.



Project Documentation Field

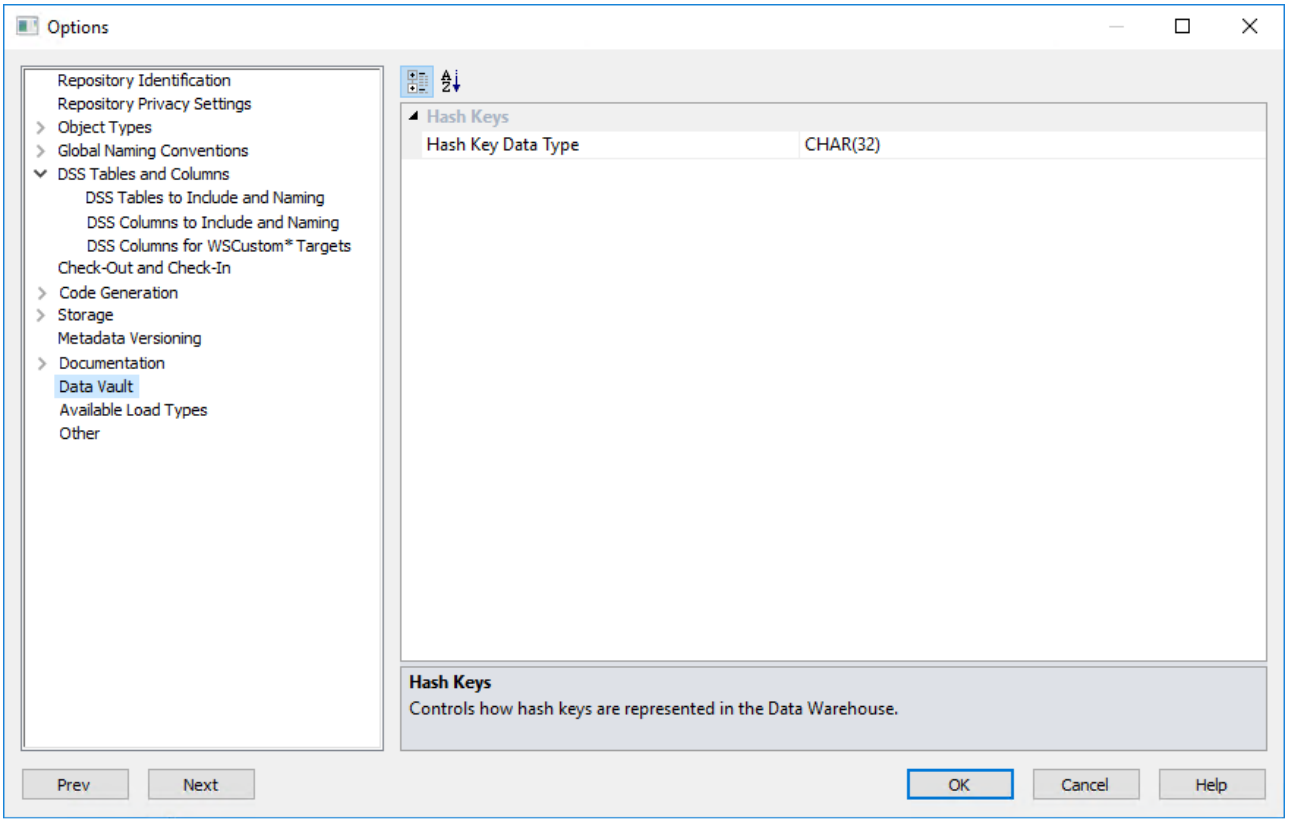
Fields	Description
Documentation Name	Sets the name of the appropriate tab in the **Properties** screen.
Documentation Label	Sets the label or description of the appropriate documentation tab.
Documentation User	Defines if the documentation information is visible to end users and included in end user documentation.
Documentation Before Columns	Defines if the documentation tab information is shown in the documentation before or after the column information.
Documentation Order	Defines the field order on the **Properties** screen tabs.

© Copyright, WhereScape Software Ltd All rights reserved.

Data Vault

This option enables you to define the data type to use for the change hash key columns generated in RED.

The generated Hash Keys are used to build the Hub, Link and Satellite objects in WhereScape RED. Refer to **Hash Key Generation Wizard** for details.



Note:

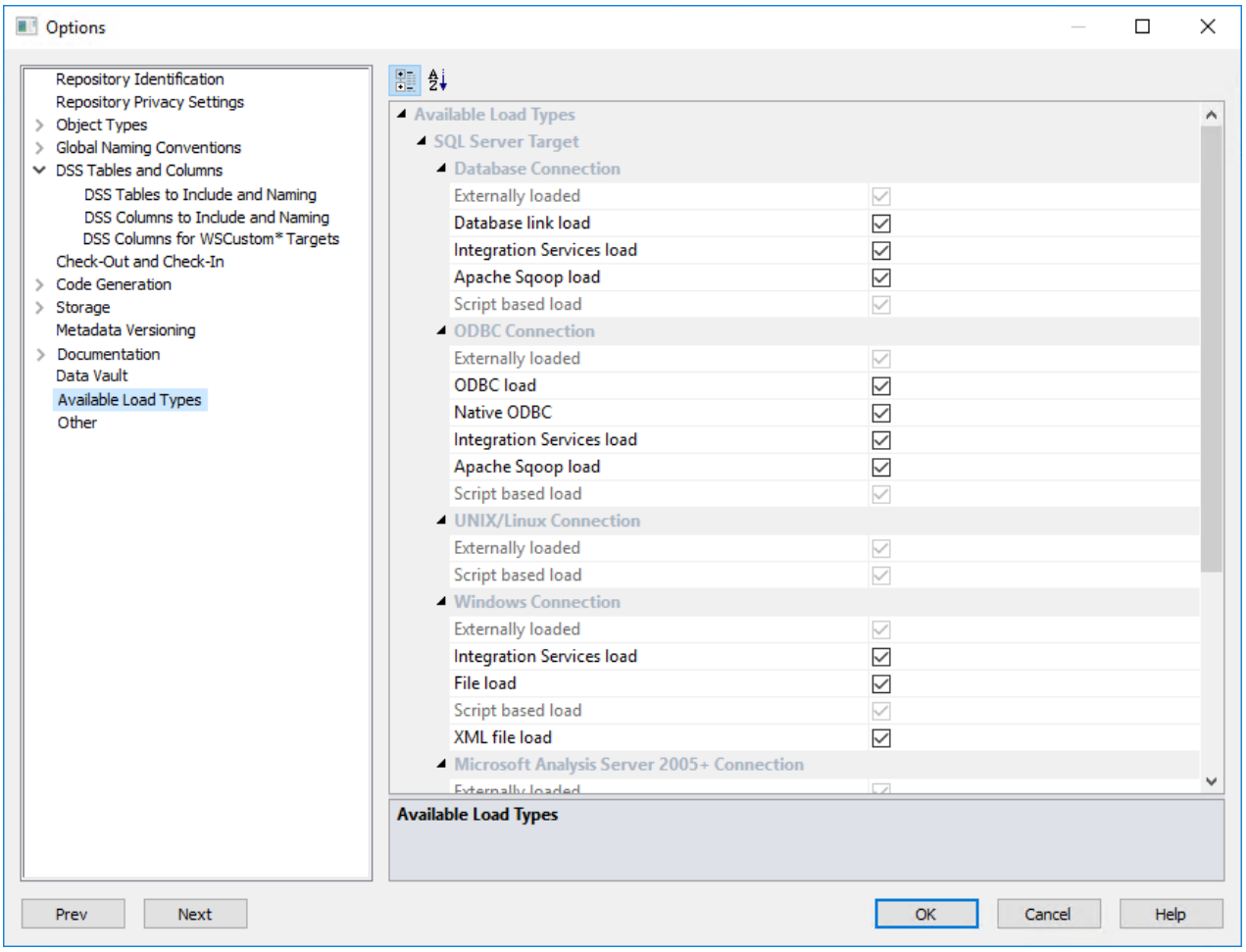
Changing this data type has no impact on any existing hash key columns.

© Copyright, WhereScape Software Ltd All rights reserved.

Available Load Types

This option enables you to activate or deactivate the various load types available in RED, based on Target and Source database.

Select from the combination of load types supported by target and source databases, to control the default load type options that can be set by users from the Connection and the Load table **Properties** windows.



Notes:

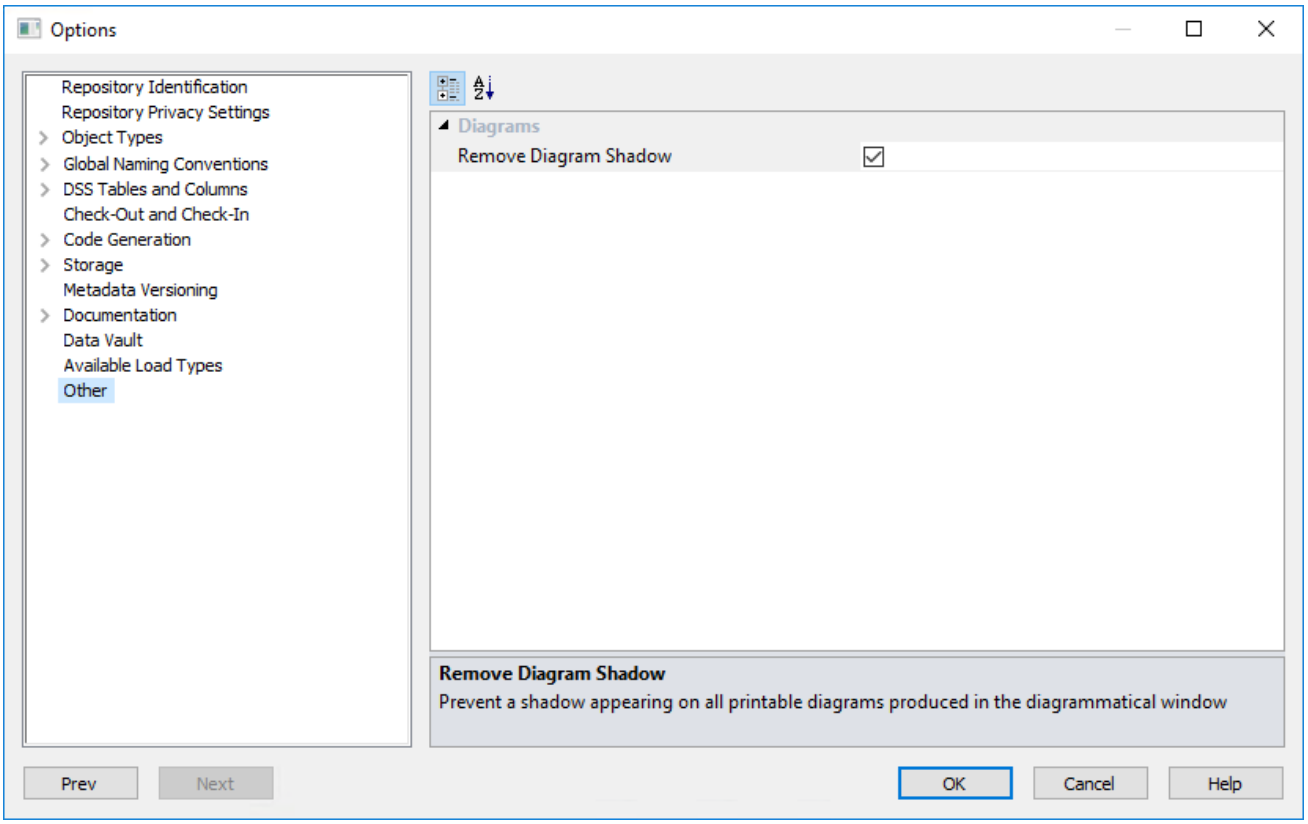
The following load types are not enabled by default for newly created **Teradata** repositories:

- Integration Services (SSIS) load in Database, ODBC and Windows connections.
- Native ODBC load in an ODBC connection.
- XML file load in a Windows connection.

© Copyright, WhereScape Software Ltd All rights reserved.

Other

This option enables users to add or remove shadows in the diagrams.



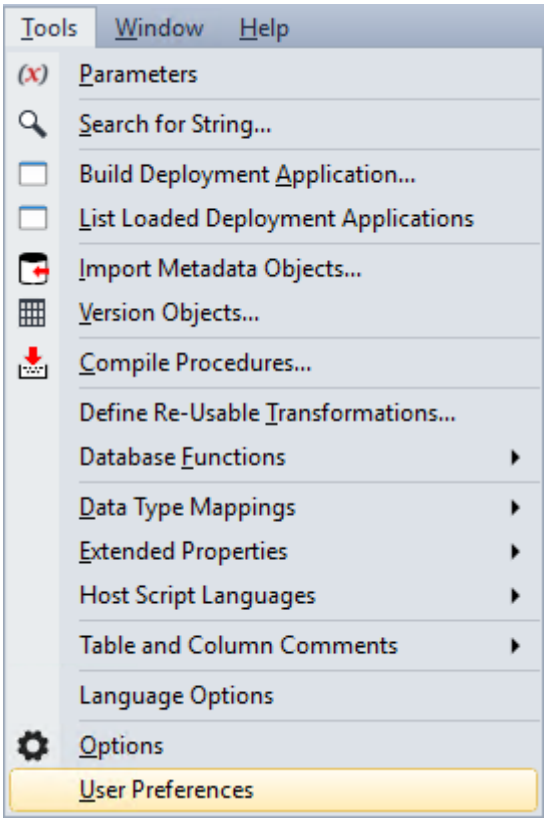
Remove Diagram Shadow

Selected by default to prevent a shadow appearing on all printable diagrams produced in the **Diagram** window; else clear to change.

© Copyright, WhereScape Software Ltd All rights reserved.

User Preferences

Select **User Preferences** from the **Tools** menu.



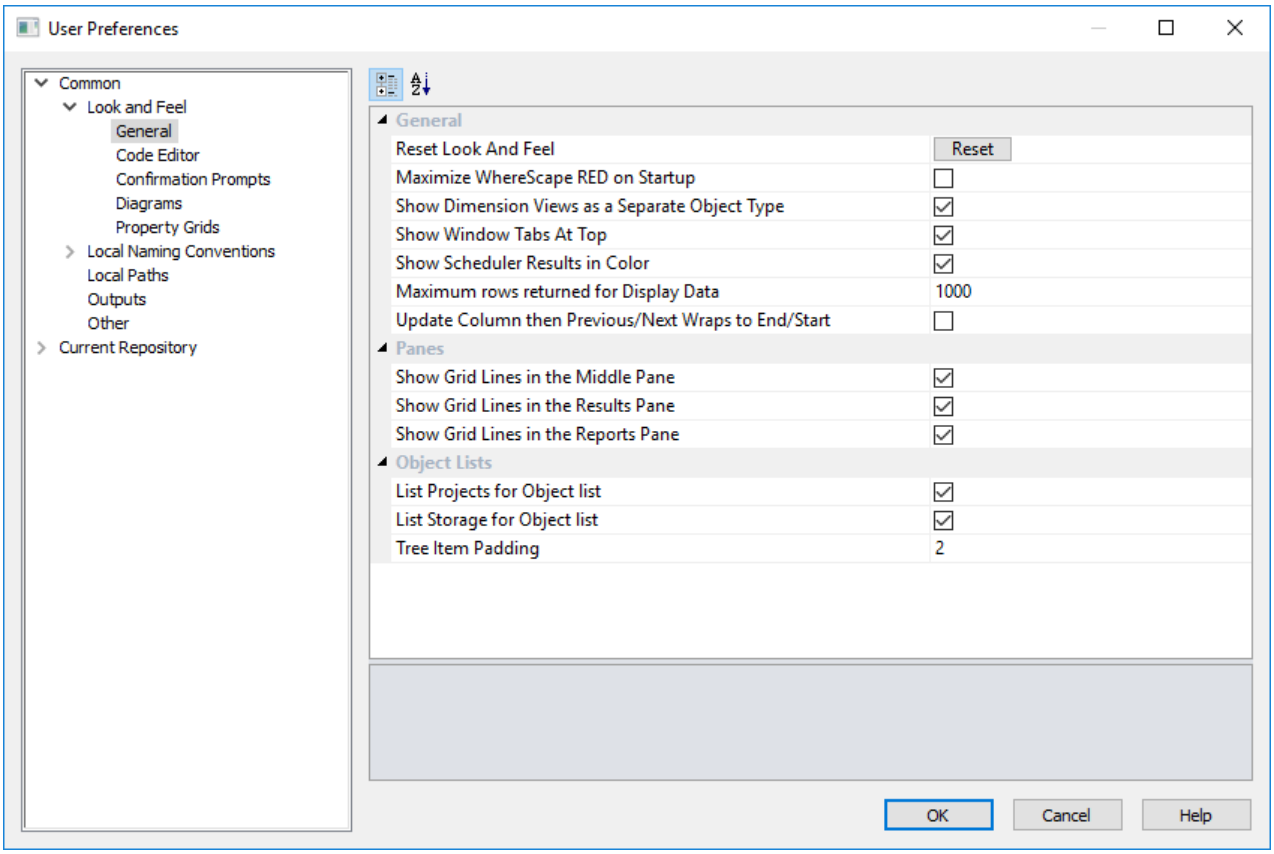
© Copyright, WhereScape Software Ltd All rights reserved.

Common

Look and Feel

General

This option enables you to set the **Look and Feel** in **General**.



General

Options	Description
Reset Look And Feel	Enables you to reset all window tab positions for Builder and Scheduler panes. Resets scheduler and report headings.
Maximize WhereScape RED on Startup	Select the check box to start WhereScape RED in full screen mode.
Show Dimension Views as a Separate Object Type	Select the check box to display a new object group called Dimension Views where all view objects are stored. This option is selected by default.
Show Window Tabs At Top	Select the check box to display window tabs at the top of the screen.
Scheduler Results in Color	Select the check box to turn on job status color coding in the scheduler. This option is selected by default.
Maximum rows returned for Display Data	This option sets the maximum number of rows that are returned when displaying data. This is set to 1000 by default.
Update Column then Previous/Next Wraps to End/Start	Select the check box to control the behavior of the directional **Update** buttons on the Column Properties window. When enabled, the **< Update** button will wrap to the last column when it moves beyond the first column; and the **Update >** button will wrap to the first column when it moves beyond the last column. When deactivated (default), the window closes after an attempt to navigate before the first column or after the last column.

© Copyright, WhereScape Software Ltd All rights reserved.

Panes

Options	Description
Show Grid Lines in the Middle Pane	Select the check box to display grid lines in the main work area.
Show Grid Lines in the Results Pane	Select the check box to display grid lines in the results area.
Show Grid Lines in the Reports Pane	Select the check ox to display grid lines in the reports area.

Object Lists

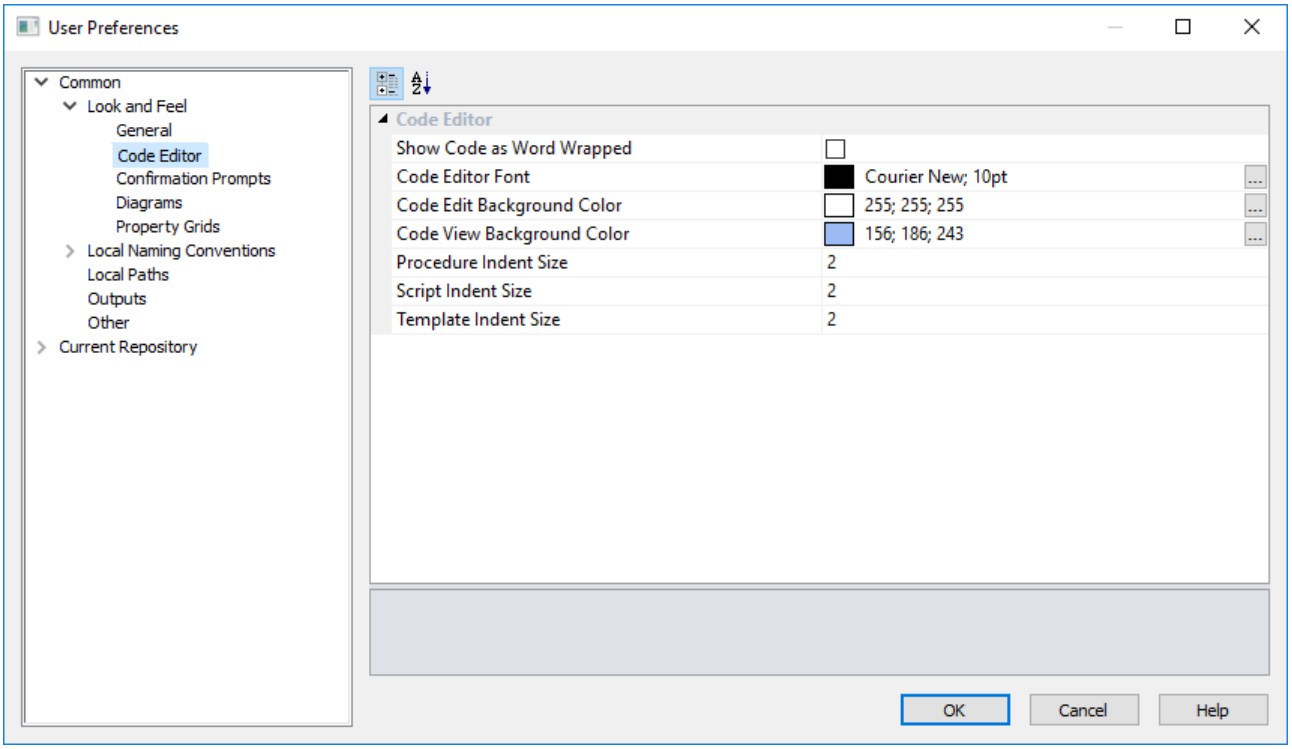
Options	Description
List Projects for Object list	Select the check box to display the projects for each object in the middle pane object list.
List Storage for Object list	Select the check box to display the storage for each object in the middle pane object list.
Tree Item Padding	Select the check box to select the number of pixels used to pad tree items when the tree items represent an Object; for example, in the Object Pane and the Browser Pane. Padding is added to the top and bottom of each tree item. Padding can be set from 0-10 pixels, default value is 2.

Note:

List Storage for Object list and **Tree Item Padding** options impact on the speed lists are generated. Since they are enabled by default, both options can be deactivated to speed up the process or if considered irrelevant according to users preferences.

Code Editor

This option enables you to set the **Look and Feel** in **Code Editor**.

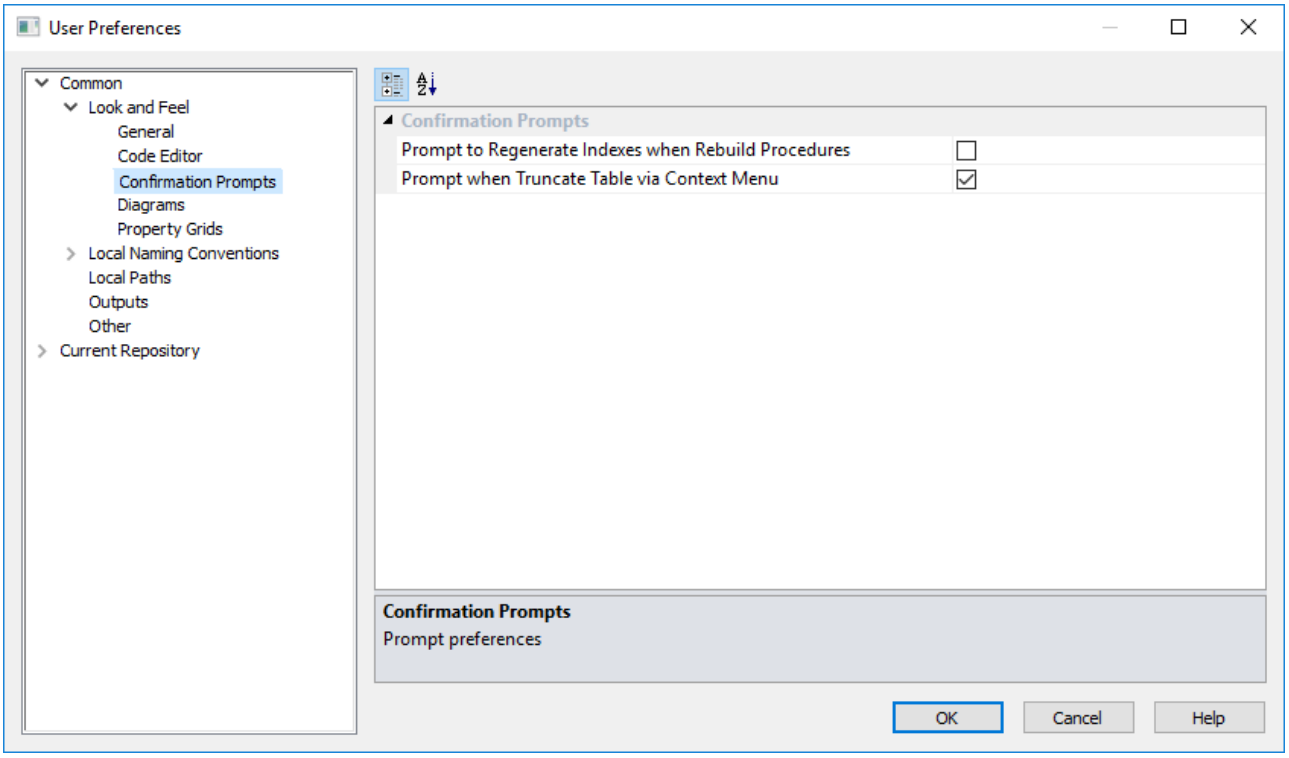


© Copyright, WhereScape Software Ltd All rights reserved.

Options	Description
Show Code as Word Wrapped	Enables you to have word wrapping applied to code by default.
Code Editor Font	Enables you to select the font used in the code editors.
Code Editor Background Color	Enables you to select the background color when editing code.
Code View Background Color	Enables you to select the background color when viewing code.
Procedure Indent Size	Enables you to specify the number of spaces that are generated when a TAB character is used within the Procedure editor. Permitted range is 2 through 10.
Script Indent Size	Enables you to specify the number of spaces that are generated when a TAB character is used within the Script editor. Permitted range is 2 through 10.
Template Indent Size	Enables you to specify the number of spaces that are generated when a TAB character is used within the Template editor. Permitted range is 2 through 10.

Confirmation Prompts

This option enables you to set the **Look and Feel** in **Confirmation Prompts**.

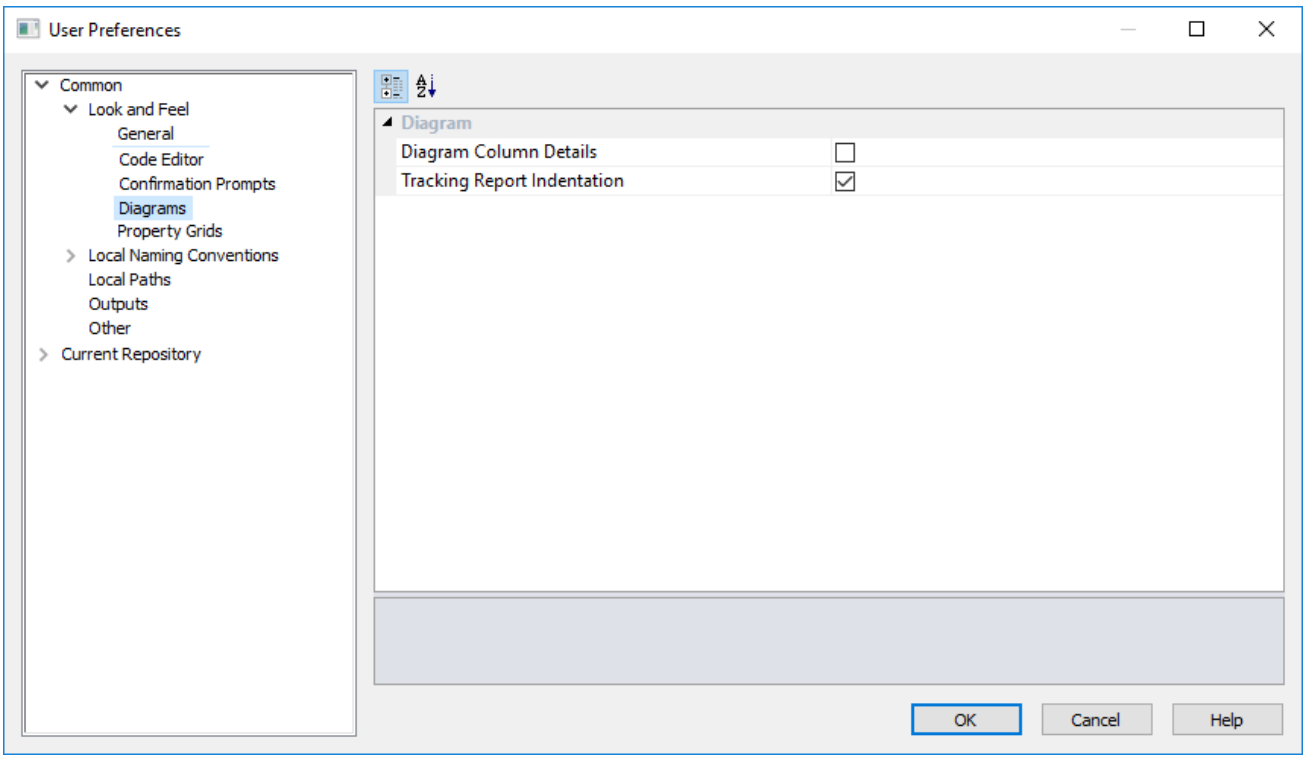


Options	Description
Prompt to Regenerate Indexes when Rebuild Procedures	When selected, always prompts for index regeneration whenever an update procedure is rebuilt.
Prompt when Truncate Table via Context Menu	When selected, always pops up a confirmation message before the truncate command is executed.

© Copyright, WhereScape Software Ltd All rights reserved.

Diagrams

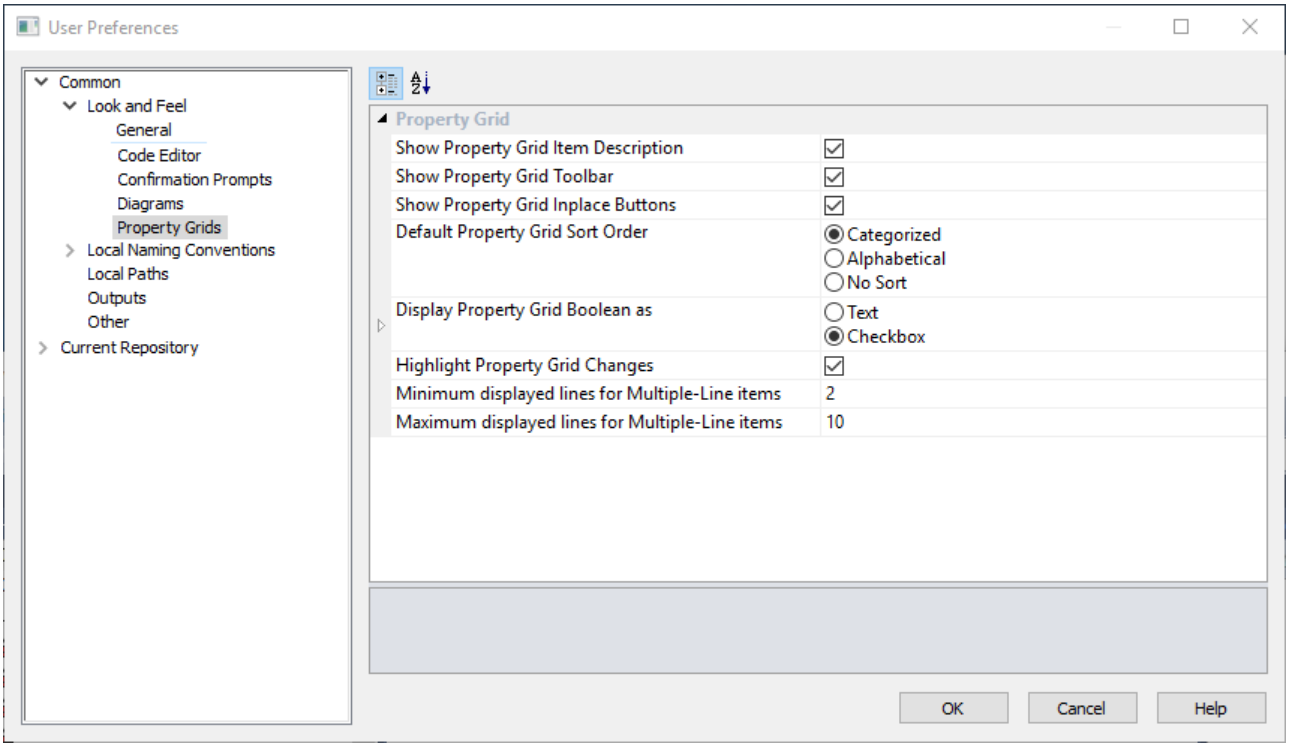
This option enables you to set the **Look and Feel** in the **Diagrams**.



Options	Description
Diagram Column Details	Select the checkbox to display the columns as the initial diagram.
Tracking Report Indentation	Select the checkbox to include tabs in the output to show dependency level.

Property Grids

This option enables you to set the **Look and Feel** in **Property Grids**.



© Copyright, WhereScape Software Ltd All rights reserved.

Options	Description
Show Property Grid Item Description	Select to display the property grid item description. The check box is selected by default.
Show Property Grid Toolbar	Select to display the property grid toolbar. The check box is selected by default.
Show Property Grid Inplace Buttons	Select to display property grid buttons for all items. The check box is selected by default.
Default Property Grid Sort Order	Enables you to select the default property grid sort order for items. The options are: • Categorized (default) • Alphabetical • No Sort
Display Property Grid Boolean as	Enables you to select how Boolean items are to be displayed. The options are: • Text (default) • Check box
Text for Boolean True	Enables you to enter the text for the Boolean value False.
Highlight Property Grid Changes	Enables you to highlight changed items in the property grid. The default is True.
Minimum displayed lines for Multiple-Line items	Sets the minimum display lines for multi-line inputs.
Maximum displayed lines for Multiple-Line items	Sets the maximum display lines for multi-line inputs.

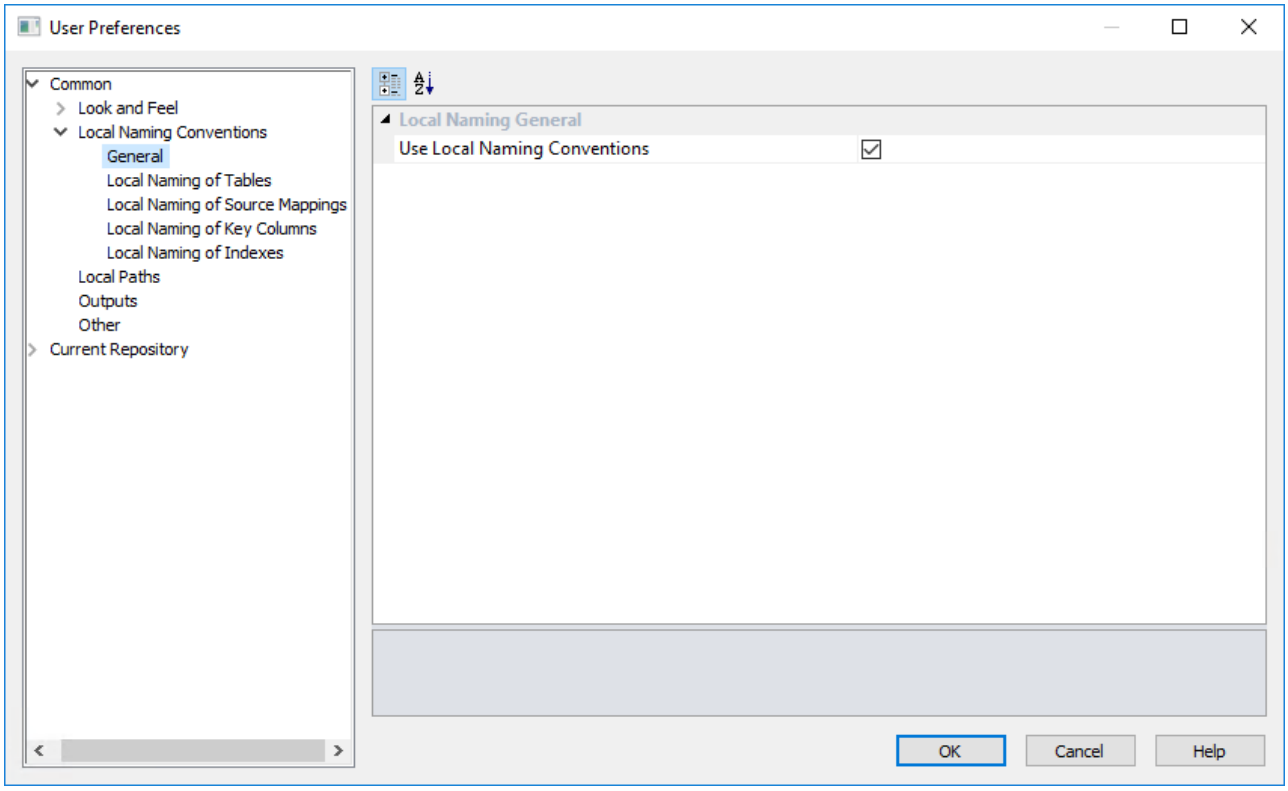
© Copyright, WhereScape Software Ltd All rights reserved.

Local Naming Conventions

The various options are described below.

General

This option enables you to set the **Local Naming Conventions**.



Options	Description
Use Local Naming Conventions	Select the checkbox to enable the **Local Naming of Tables**, **Key Columns** and **Indexes** options in the object tree.

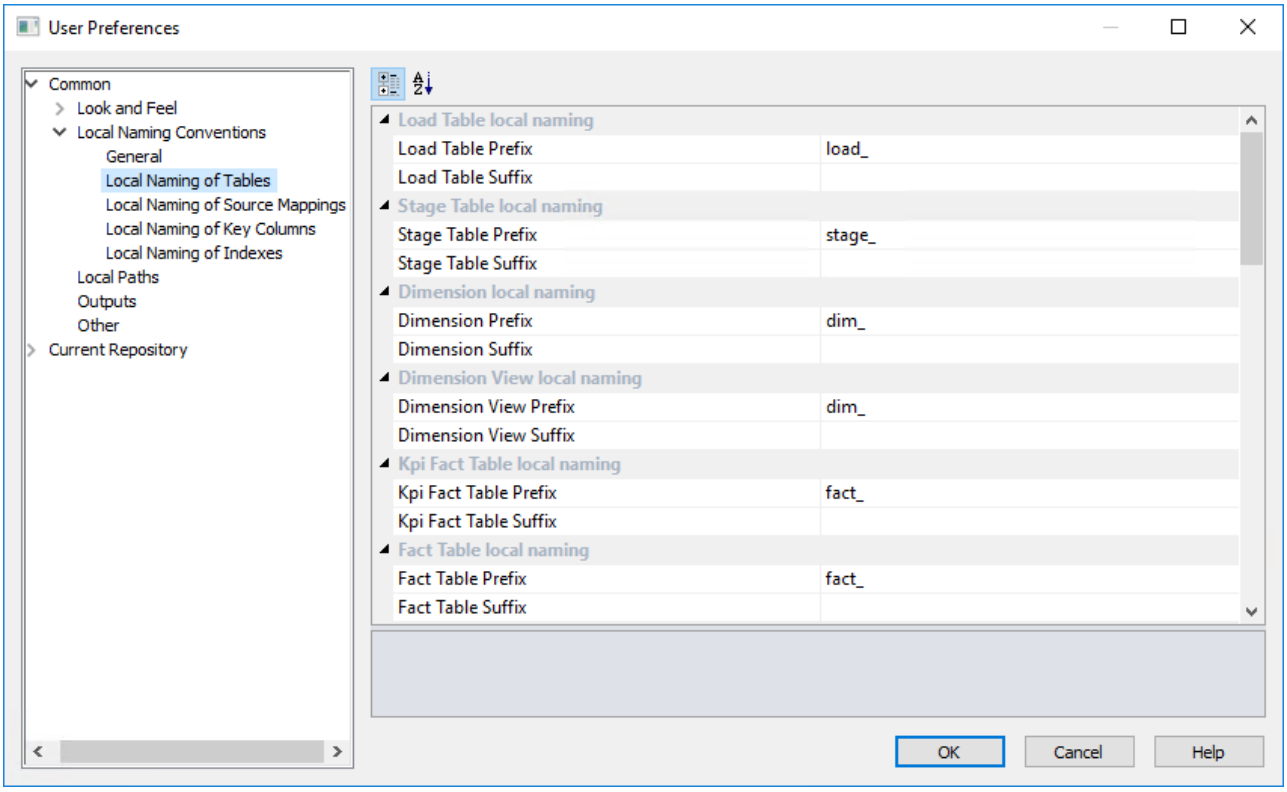
Note

If this option is set, it can overwrite short names and object prefixes.

© Copyright, WhereScape Software Ltd All rights reserved.

Local Naming of Tables

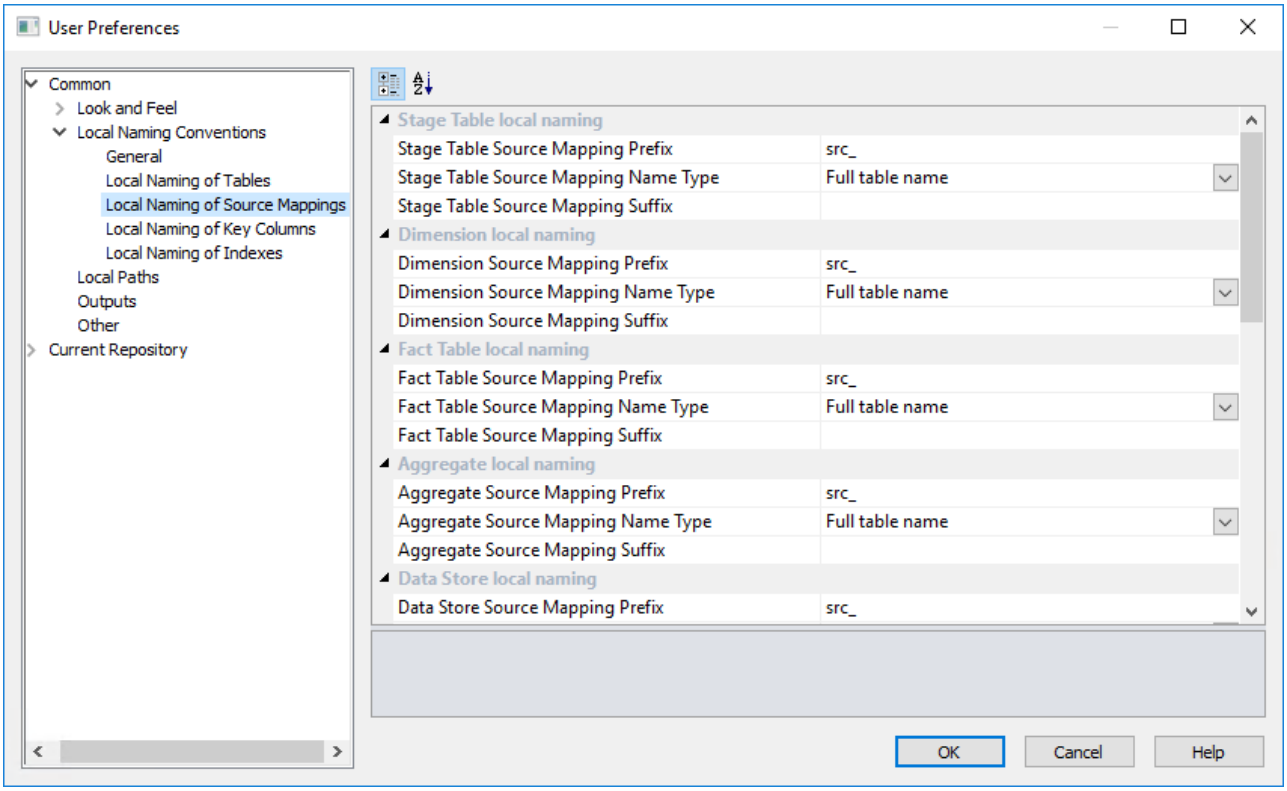
This option enables you to set the **Local Naming of Tables**.



Define the **prefix** and **suffix** used in the default naming convention for each table type.

Local Naming of Source Mapping

This option enables users to set the **Local Naming of Source Mappings** options.

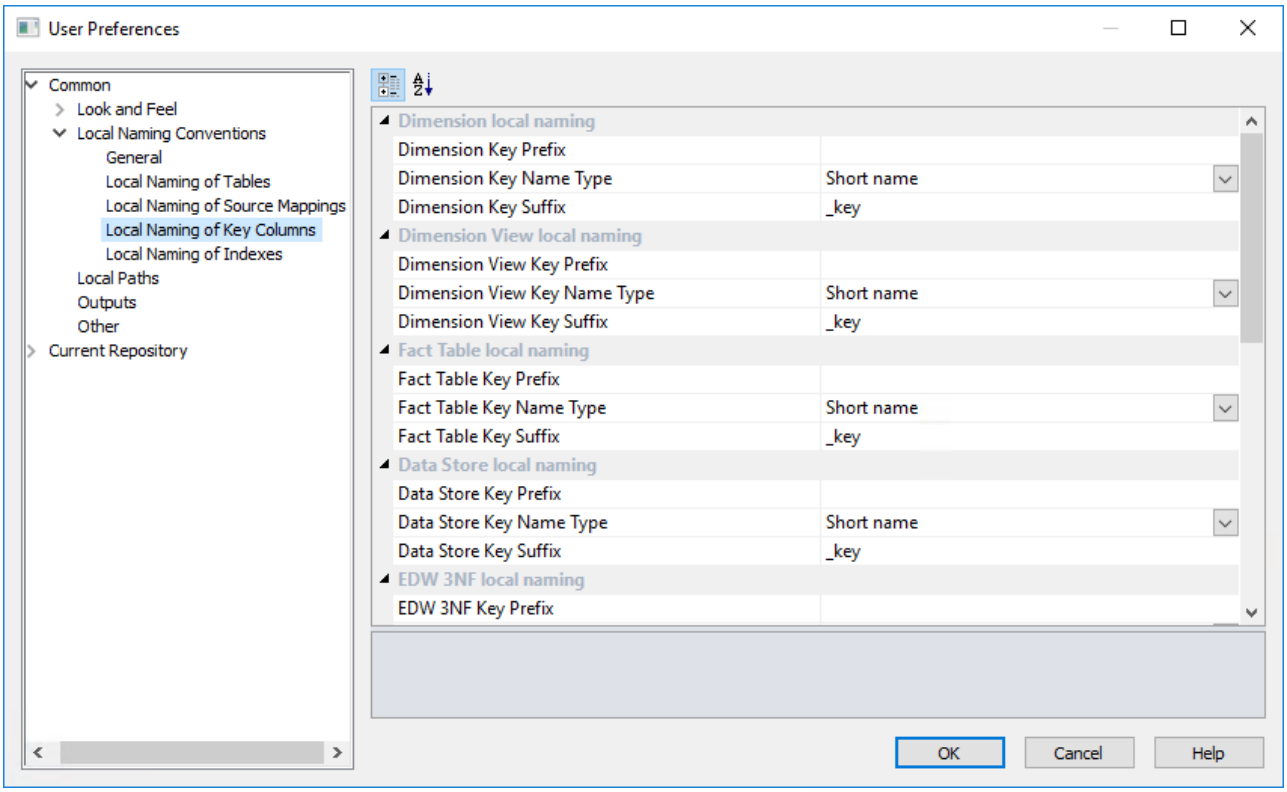


© Copyright, WhereScape Software Ltd All rights reserved.

Options	Description
Source Mapping Prefix	Enables you to set the prefix used in the default source mapping naming convention.
Source Mapping Name Type	Enables you to set the basis for the source mapping naming.
Source Mapping Suffix	Enables you to set the suffix used in the default source mapping naming convention.

Local Naming of Key Columns

This option enables you to set the **Local Naming of Key Columns**.

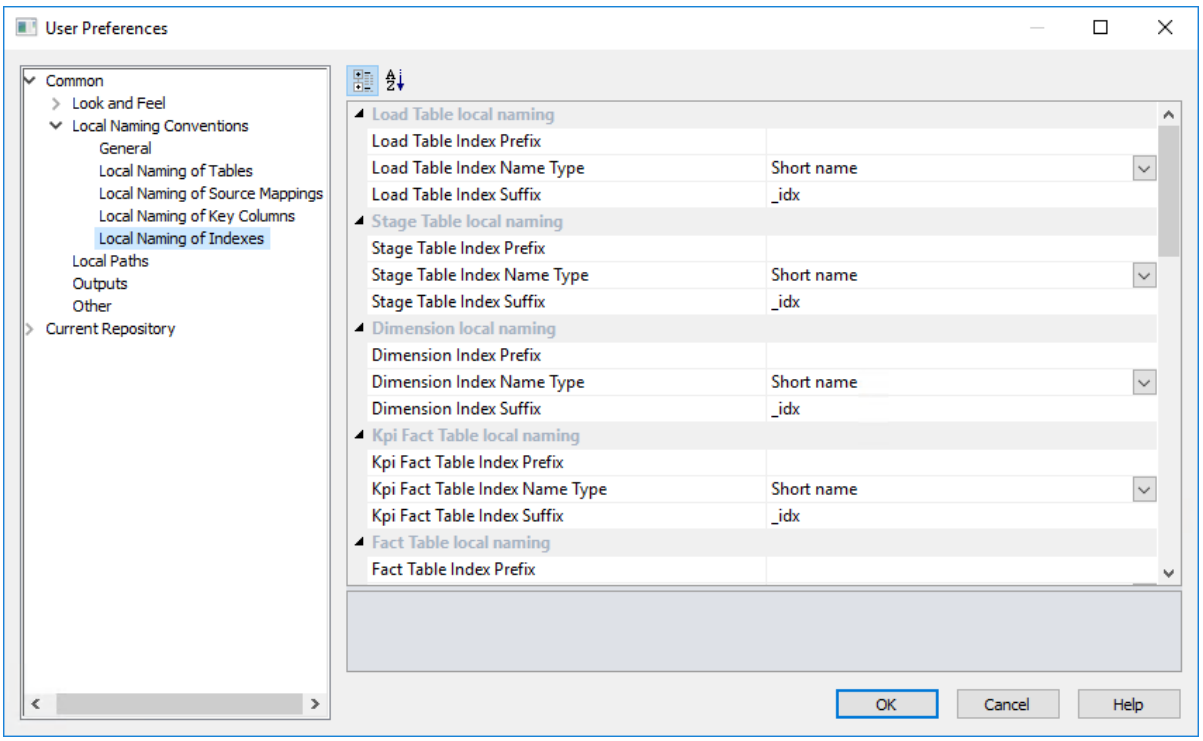


Options	Description
Key Prefix	Sets the prefix used in the default key naming convention.
Key Name Type	Sets the basis for the key naming.
Key Suffix	Sets the suffix used in the default naming convention.

© Copyright, WhereScape Software Ltd All rights reserved.

Local Naming of Indexes

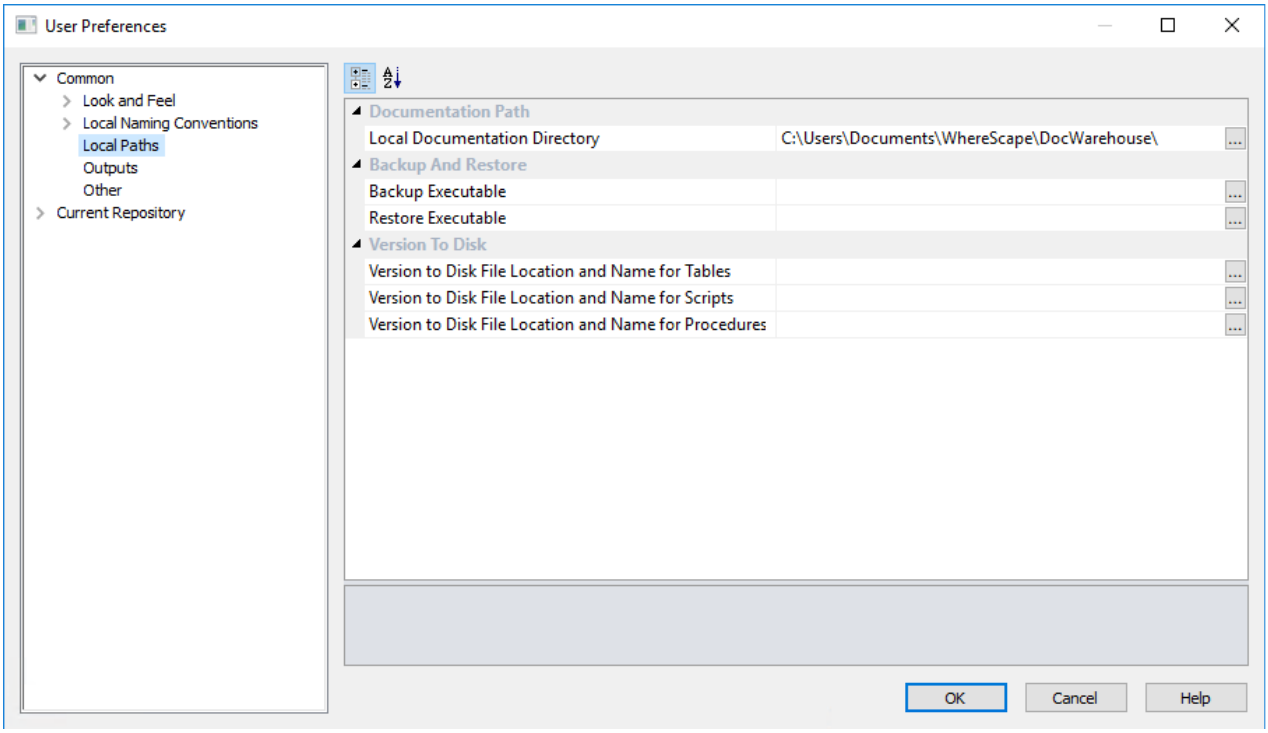
This option enables you to set the **Local Naming of Indexes**.



Options	Description
Index Prefix	Enables you to set the prefix used in the default index naming convention.
Index Name Type	Enables you to set the basis for the index naming.
Index Suffix	Enables you to set the suffix used in the default index naming convention.

Local Paths

This option enables you to set the **Local Paths** for documentation, backup and restore and for versioning to disk.



© Copyright, WhereScape Software Ltd All rights reserved.

Documentation Path

Set the **Local Documentation Directory**.

Options	Description
Local Documentation Directory	Sets the Local Documentation Directory

Backup And Restore

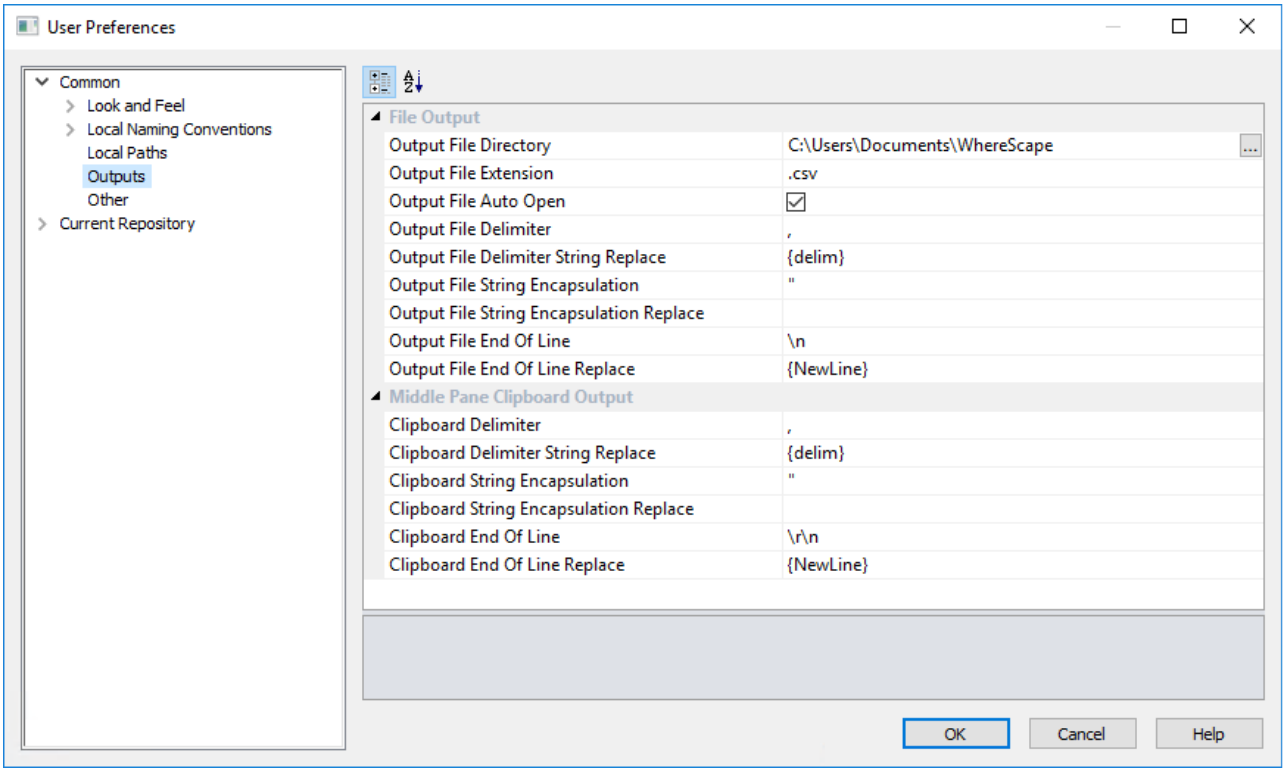
Options	Description
Backup Executable	Sets the override for backup executable. By default WhereScape RED tries to find the path of the backup executable. This is bcp.exe for SQL Server, exp.exe for Oracle and db2cmd.exe for IBM DB2. This edit box provides the ability to specify the exact location and name of the executable. This is useful when WhereScape RED cannot find the program or if there are multiple versions of the program on the PC.
Restore Executable	Sets the override for restore executable. By default, WhereScape RED tries to find the path of the restore executable. This is bcp.exe for SQL Server, imp.exe for Oracle and db2cmd.exe for IBM DB2. This edit box provides the ability to specify the exact location and name of the executable. This is useful when WhereScape RED cannot find the program or if there are multiple versions of the program on the PC.

Version to Disk

Options	Description
Versions to Disk options	Sets the locations and names. If any of the three version to disk paths are set, WhereScape RED automatically creates ASCII files containing the applicable DDL or code, each time an automated version occurs in the entered directory.

Outputs

This option enables you to set the **Output** user preferences.



© Copyright, WhereScape Software Ltd All rights reserved.

File Output

Options	Description
Output File Directory	Enables you to set the path for output files created from the middle pane.
Output File Extension	Select the checkbox to set the file extension for output files created from the middle pane. This value determines the program that will auto open files.
Output File Auto Open	Select the checkbox to automatically open the results in files created from the middle pane.
Output File Delimiter	Enables you to set the characters that separate each field within each record of output files created from the middle pane. Common values are , and |.
Output File Delimiter String Replace	Enables you to set the characters that will replace the delimiter character if it occurs inside a field.
Output File String Encapsulation	Enables you to set the characters that are used to enclose string values of files created from the middle pane. Common values are " and '.
Output File String Encapsulation Replace	Enables you to set the characters that will replace the encapsulation string if it occurs inside a field.
Output File End Of Line	Enables you to set the characters saved at the end of each record of files created from the middle pane. Common values are \n, \r and \t.
Output File End Of Line Replace	Enables you to set the characters that will replace the end of line string if it occurs inside a field.

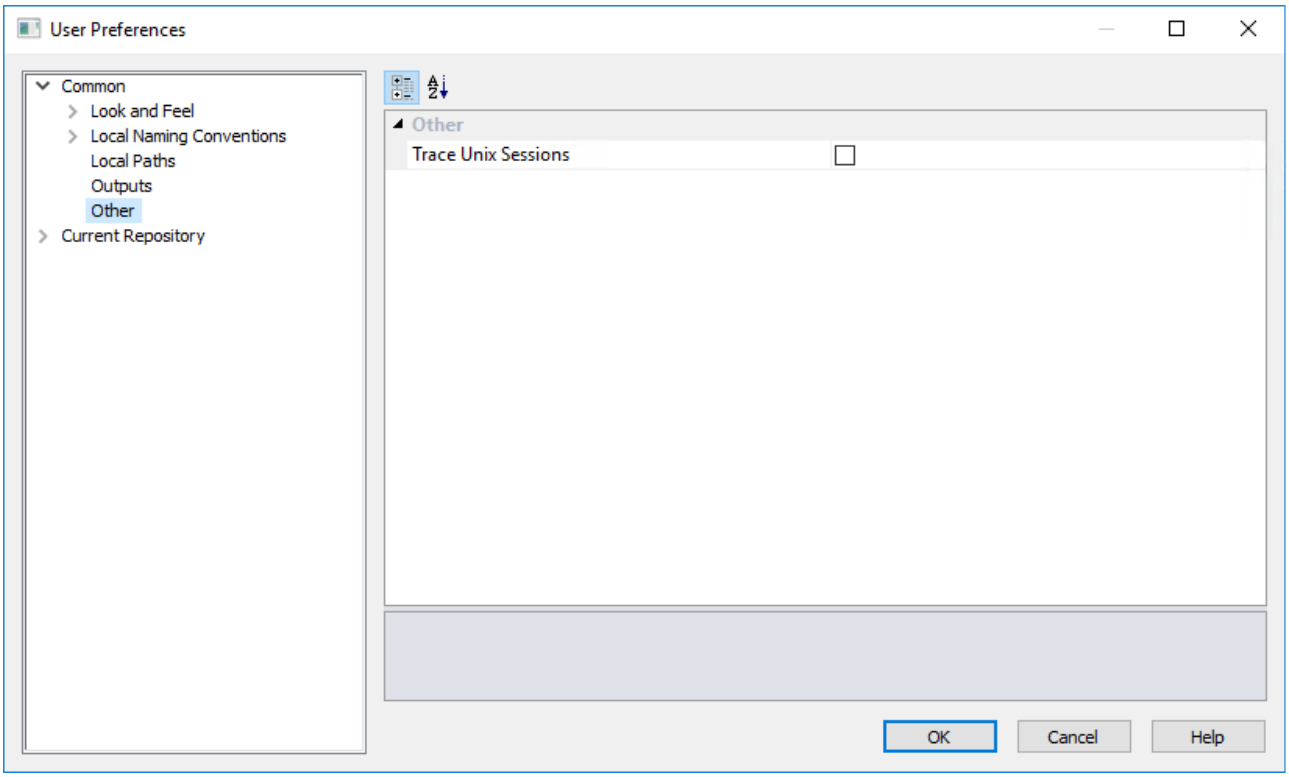
Middle Pane Clipboard Output

Options	Description
Clipboard Delimiter	Enables you to set the characters that separate each field within each record of clipboard output created from the middle pane. Common values are , and |.
Clipboard Delimiter String Replace	Enables you to set the characters that will replace the delimiter character if it occurs inside a field.
Clipboard String Encapsulation	Enables you to set the characters that are used to enclose string values of clipboard output created from the middle pane. Common values are " and '.
Clipboard String Encapsulation Replace	Enables you to set the characters that will replace the encapsulation string if it occurs inside a field.
Clipboard End of Line	Enables you to set the characters saved at the end of each record of clipboard output created from the middle pane. Common values are \n, \r and \t.
Clipboard End of Line Replace	Enables you to set the characters that will replace the end of line string if it occurs inside a field.

© Copyright, WhereScape Software Ltd All rights reserved.

Other

This option enables you to set the **Other** user preferences.



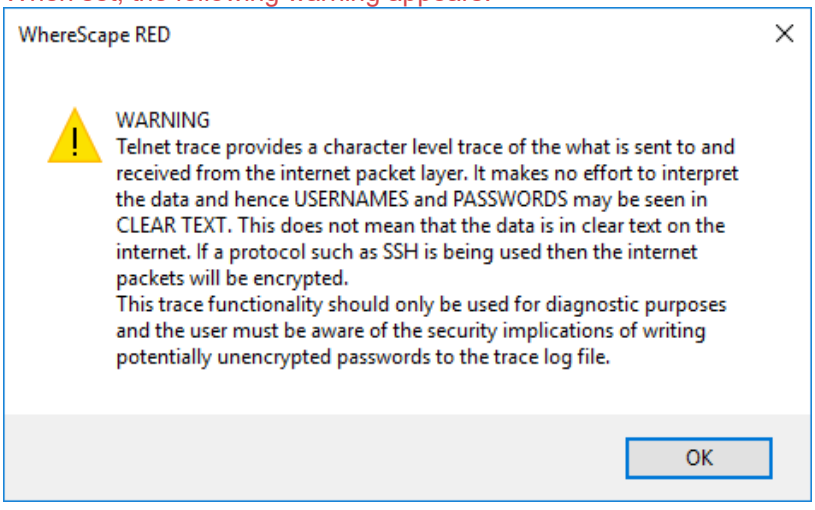
Options	Description
Trace Unix Sessions	Select the checkbox to trace all Unix activity undertaken by the WhereScape RED program until it is terminated. The file WslMedTelnet.txt is created in the program directory for WhereScape RED.

Tip

This option is intended for the debugging of specific Unix problems and the setting of this switch would normally be done at the request of WhereScape when attempting to solve a Telnet issue. This setting is only relevant for the PC on which the setting is made (e.g. it is not a global setting for the repository).

Note

When set, the following warning appears:

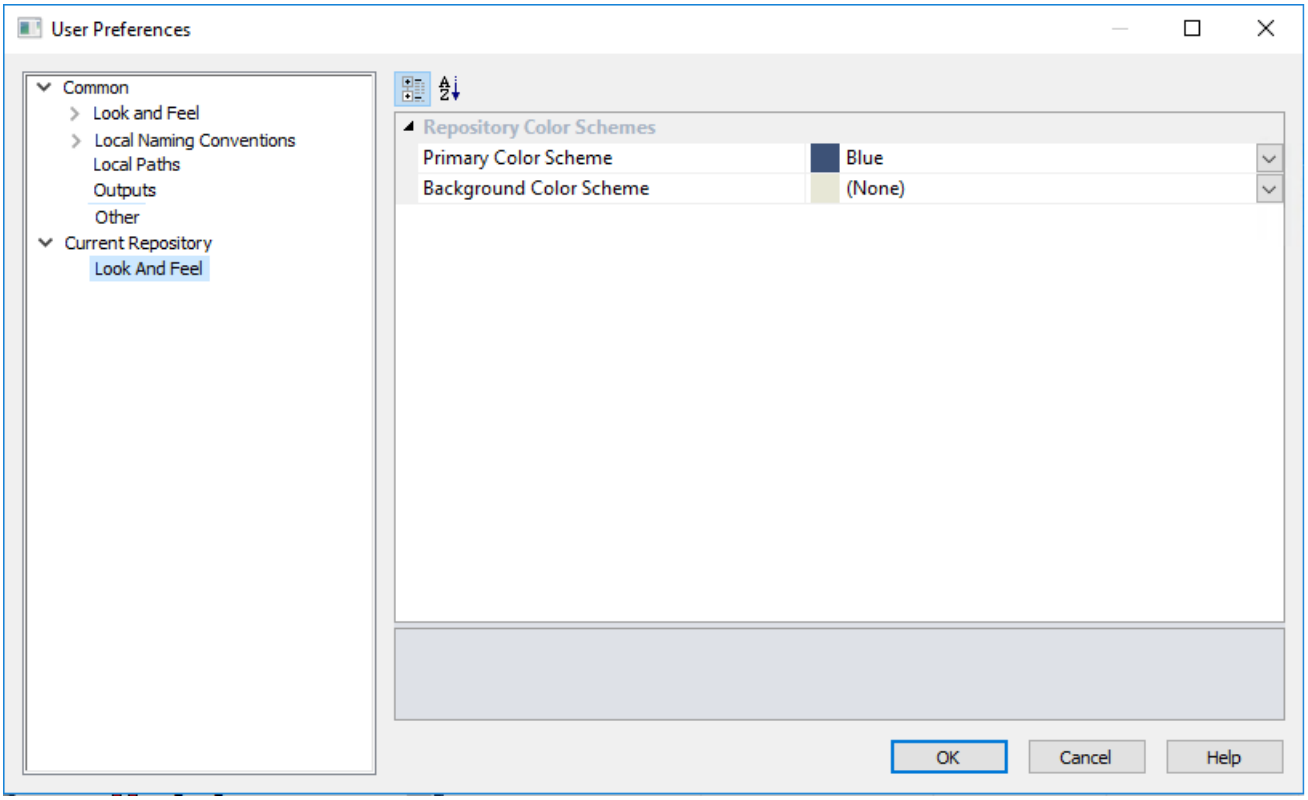


© Copyright, WhereScape Software Ltd All rights reserved.

Current Repository

Look and Feel

This option enables you to set the **Look and Feel** for the **Current Repository**.

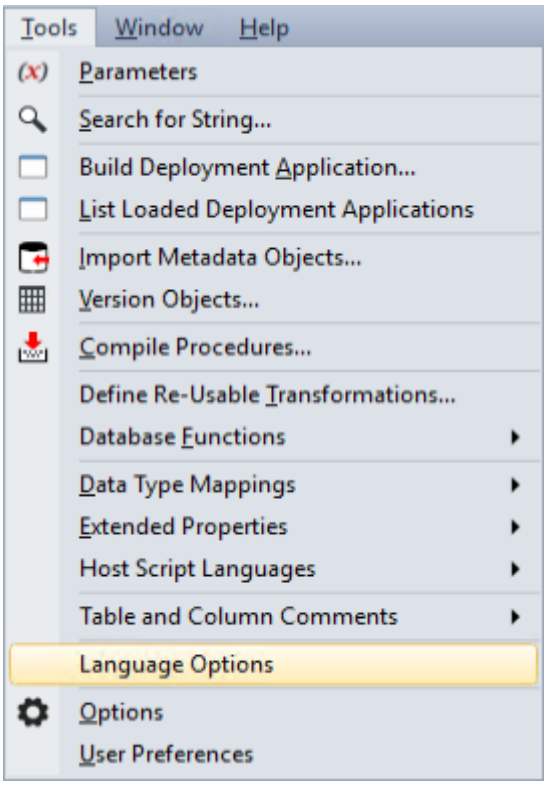


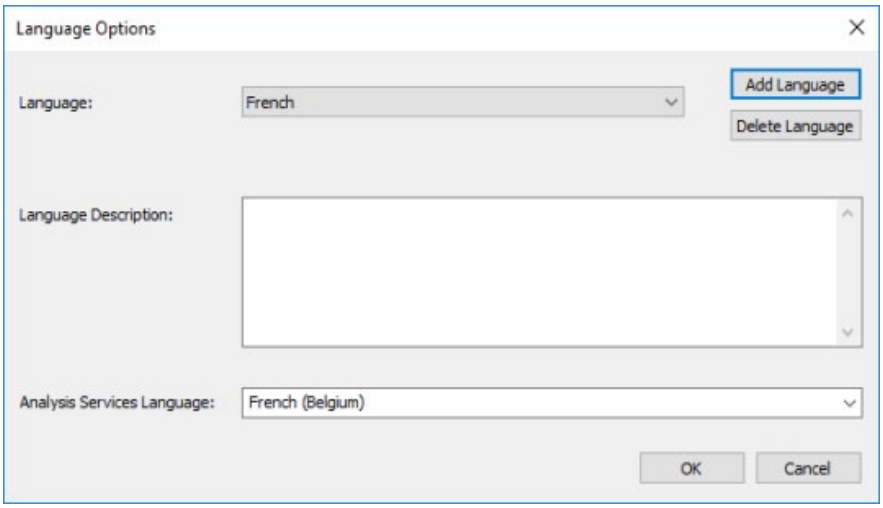
Options	Description
Repository Color Scheme	Enables you to set the primary and background **Color Schemes** for the current repository.

© Copyright, WhereScape Software Ltd All rights reserved.

Language Options

This option is only available for SQL Server databases and applies only to dimension, fact and OLAP objects.
A blank entry means that no languages have been defined and thus, no translations can be saved.
Select **Language Options** from the **Tools** menu.



Options	Description
To add a language	Click the **Add Language** button; enter the new language ID and then click **OK**.
To delete a language	First select the language from the drop-down list and then click the **Delete Language** button. **Note:** All translations for the selected language is also deleted. 
Language	The language Reference/ID.
Language Description	The language description.

© Copyright, WhereScape Software Ltd All rights reserved.

Options	Description
Analysis Services Language	Used to identify the language ID, as used in Analysis Services.

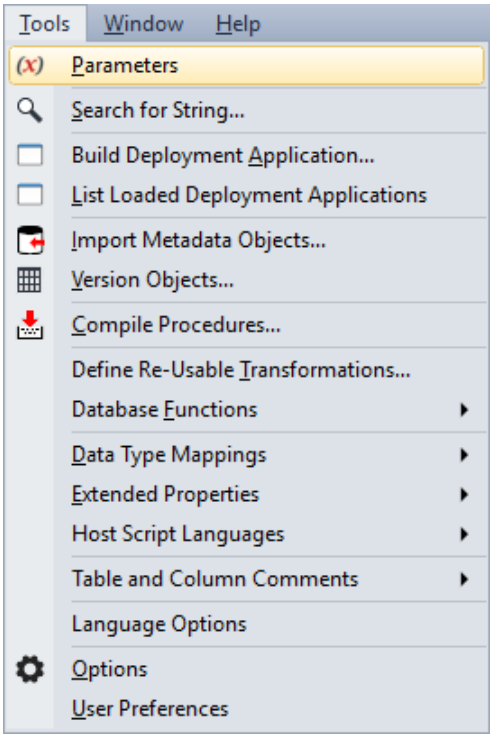
© Copyright, WhereScape Software Ltd All rights reserved.

Parameters

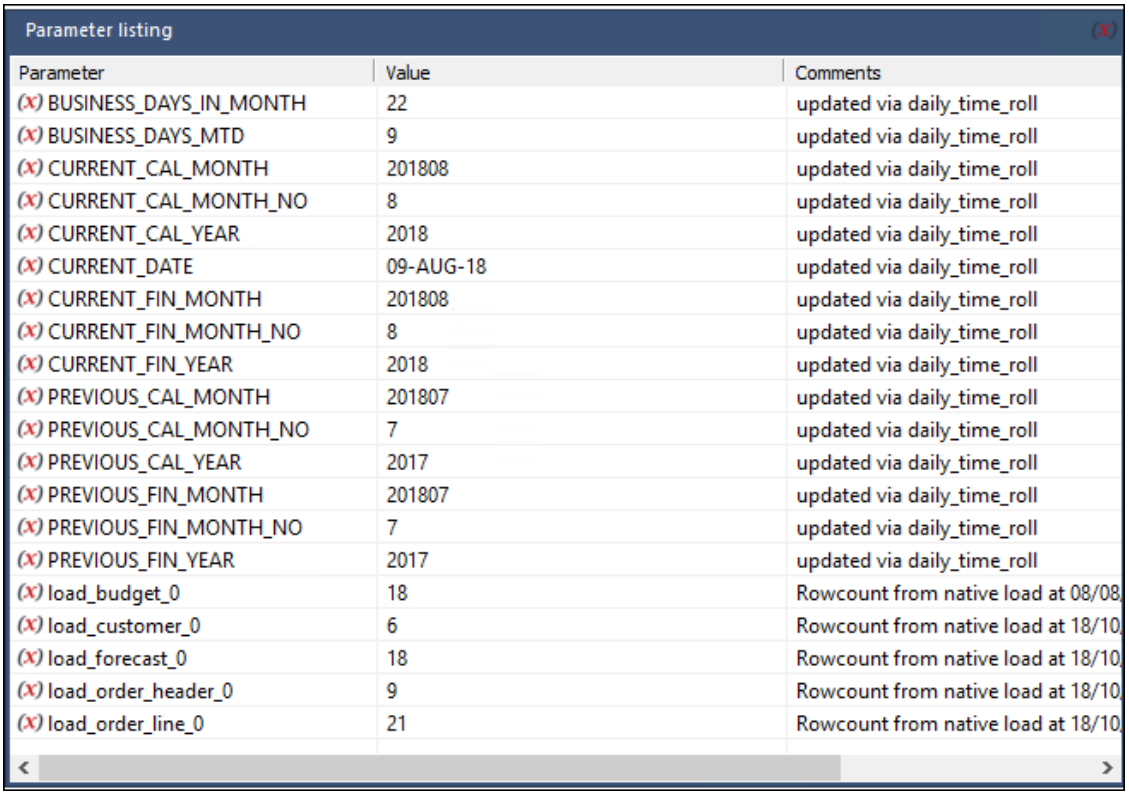
Parameters are a means of passing information between two or more procedures and between the WhereScape RED environment and procedures.

They can be accessed within the WhereScape RED environment in two ways:

1. **Tools > Parameters** menu option:



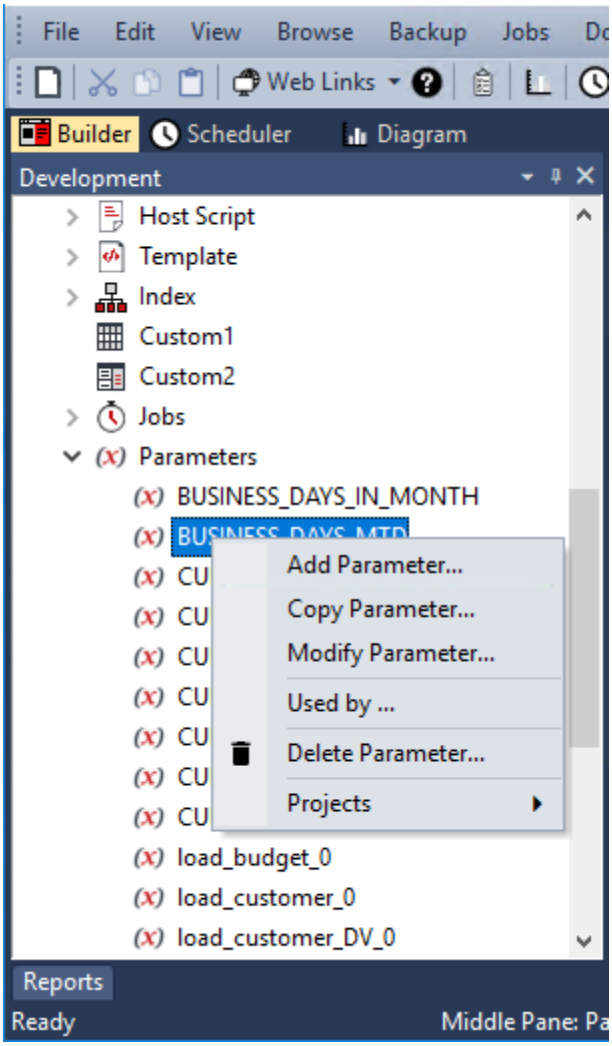
A list of parameters is displayed as per the example below:



Parameter	Value	Comments
BUSINESS_DAYS_IN_MONTH	22	updated via daily_time_roll
BUSINESS_DAYS_MTD	9	updated via daily_time_roll
CURRENT_CAL_MONTH	201808	updated via daily_time_roll
CURRENT_CAL_MONTH_NO	8	updated via daily_time_roll
CURRENT_CAL_YEAR	2018	updated via daily_time_roll
CURRENT_DATE	09-AUG-18	updated via daily_time_roll
CURRENT_FIN_MONTH	201808	updated via daily_time_roll
CURRENT_FIN_MONTH_NO	8	updated via daily_time_roll
CURRENT_FIN_YEAR	2018	updated via daily_time_roll
PREVIOUS_CAL_MONTH	201807	updated via daily_time_roll
PREVIOUS_CAL_MONTH_NO	7	updated via daily_time_roll
PREVIOUS_CAL_YEAR	2017	updated via daily_time_roll
PREVIOUS_FIN_MONTH	201807	updated via daily_time_roll
PREVIOUS_FIN_MONTH_NO	7	updated via daily_time_roll
PREVIOUS_FIN_YEAR	2017	updated via daily_time_roll
load_budget_0	18	Rowcount from native load at 08/08
load_customer_0	6	Rowcount from native load at 18/10
load_forecast_0	18	Rowcount from native load at 18/10
load_order_header_0	9	Rowcount from native load at 18/10
load_order_line_0	21	Rowcount from native load at 18/10

© Copyright, WhereScape Software Ltd All rights reserved.

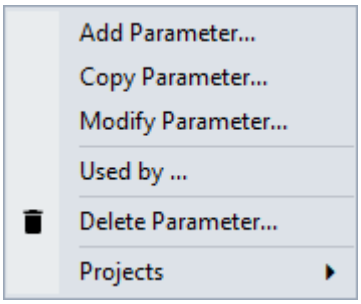
2. **Objects pane > Parameter** objects list:



© Copyright, WhereScape Software Ltd All rights reserved.

Managing Parameters

A parameter can be added, edited, copied or deleted using the right-click context menu of the Parameter column:



Parameters can also be added or removed from Projects you have defined in RED using the **Projects** context menu option. Refer to **Organizing Objects** for details .

Parameter Usage

Typical parameter usage may be the global definition of how many days should be looked back for changed data, a month or processing period, etc.

Parameters can be used in **Load** tables to place limits in a 'Where' clause, etc. Refer to **Database Link Load - Source Mapping** for details.

They are also used by **Stage table procedures** as variables. Refer to **Generating the Staging Update Procedure** for details.

- **Oracle**: Two functions are provided to allow procedures to read and write parameters. These functions are **WsParameterRead** and **WsParameterWrite**. Using these functions a procedure can load and use the contents of a parameter, or modify an existing parameter, or add a new parameter.
- **SQL Server**: Two procedures are provided to allow procedures to read and write parameters. These procedures are **WsParameterRead** and **WsParameterWrite**. Using these procedures, a procedure can load and use the contents of a parameter, or modify an existing parameter, or add a new parameter. A function **WsParameterReadF** is also provided. This function will return the value of a parameter.

Global Parameters

A number of global parameters are provided when loading tables, either via WhereScape RED or via the scheduler. These global parameters are accessed by calling the function **WsParameterReadG**.

The Load table name and the source table name can be acquired by using this function.

Refer to **Callable Routines** for more information on **WsParameterRead**, **WsParameterWrite**, **WsParameterReadF** and **WsParameterReadG**.

© Copyright, WhereScape Software Ltd All rights reserved.

Connections

Connection objects serve several purposes in WhereScape RED:

1. They are used to browse potential source data in source systems and to acquire metadata. Potential source data includes database tables and flat files.

   For **database tables**, WhereScape RED:

   - Uses the **ODBC Source** set on each connection to browse the source system.
   - Acquires the metadata for new **load tables** built from the source system using drag and drop.

   For **files**, WhereScape RED:

   - Connects directly to Windows, UNIX/Linux and Hadoop to analyze the source file for the new load table and to acquire its metadata.
   - Prompts for user input for any metadata not available in the source file.

   Notes:

   - ODBC connections must be either **User DSN** or **System DSN**. **File DSN** connections are **not** supported.
   - **Windows** and **UNIX** connections do not have an **ODBC Source** property. **UNIX** connections are used for UNIX and Linux systems.

2. Load tables with a connection of **Connection type ODBC** extract data from source systems using ODBC. The **ODBC Source** of the connection is the ODBC DSN used for the extract.

   Note:

   In Teradata environments, if a **Teradata TPT compliant ODBC DSN** is defined in the **Connection Properties**, the **TPT DSN** is used for TPT ODBC Loads.

3. Each data warehouse metadata repository must have a Data Warehouse connection to use drag and drop to create new objects (other than load tables) in the data warehouse. WhereScape RED:
   - Uses the **ODBC Source** set on the Data Warehouse connection to browse the Data Warehouse database.
   - Acquires the metadata for any **tables** built from existing data warehouse tables.

   Note:

   This connection always has a **Connection type** of **Database**.

4. Cube objects require a connection to define the Analysis Services server used to create and load cubes. This is a connection with a **Connection type** of **Microsoft Analysis Server 2005+.**
5. Export objects require a connection to define the target environment where exported data is written. This is a connection with a **Connection type** of **UNIX** or **Windows.**

© Copyright, WhereScape Software Ltd All rights reserved.

Connection Types

Connection properties are described in this topic as they apply to the supported database types in WhereScape RED.
Connection types can be set up via the following methods:

- Connection to the Data Warehouse/Metadata repository - Database type connections
- Connections to another Database - Database type connections on the same server or through a linked server
- ODBC Based Connections - ODBC type connections with Native ODBC, ODBC, Externally loaded or Integration Services load types
- Connections to Windows
- Connections to UNIX/Linux
- Connections to Hadoop
- Connections to Microsoft Analysis Services

One or more of the following icons are used to indicate where properties are unique to a particular database type:

Note

Custom type database connections are assigned a unique name during the licensing process. In this document, the term 'Custom' refers to all custom database connections.

Database - Data Warehouse/Metadata Repository

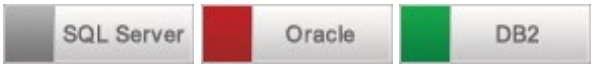
This topic describes the details of the connection Properties as they apply to **Database** type connections and specifically of the Data Warehouse or Metadata repository connection.
The Data Warehouse connection is the connection that stores the metadata repository and it is the connection that is used in the drag and drop functionality to create the Dimension, Stage, Fact and Aggregate tables.

This connection is also used to create Cubes.

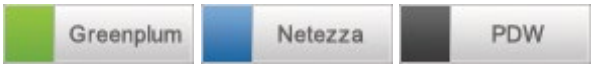
Connection types also impact the available load methods.

Note:

The **Data Warehouse connection** must exist if you wish to use drag and drop to create Dimensions, Stage tables, Fact tables, Aggregates and Cubes.

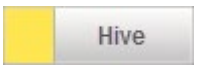


SQL Server, Oracle and **DB2 Data Warehouse** - the connection for the Metadata repository.



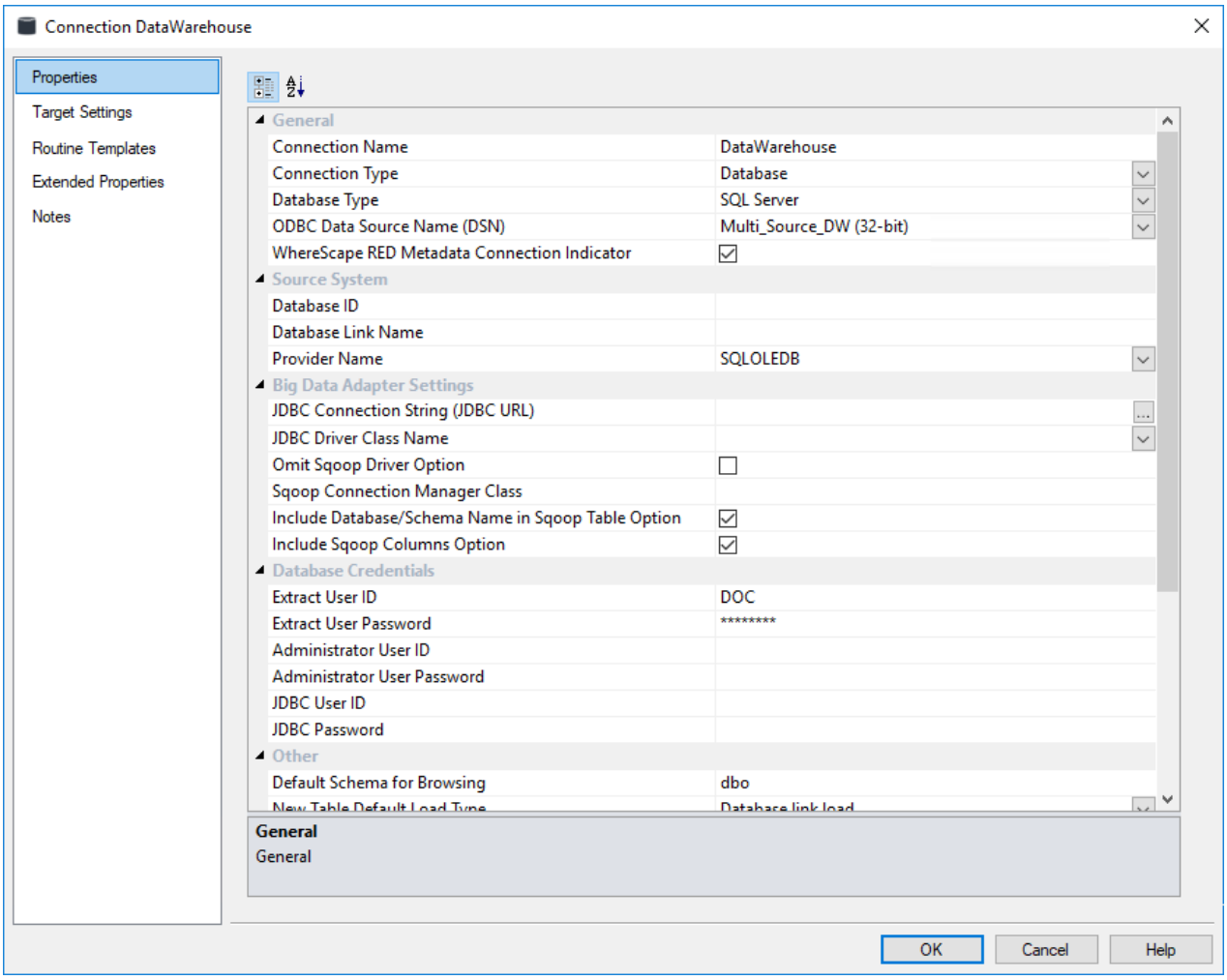
Greenplum, Netezza and PDW Data Warehouse - the connection for the Metadata repository which is always stored in **SQL Server**.

Apart from the Data Warehouse or Repository connection to SQL Server, there must be at least one other connection to the Target Data warehouse database, such as Greenplum, Netezza or PDW.



Hive can have its metadata repositories in Oracle, SQL Server and Teradata data warehouses.

© Copyright, WhereScape Software Ltd All rights reserved.



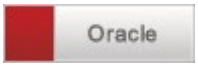
General

Options	Description
Connection Name	Name used to label the connection within WhereScape RED. Typically for SQL Server, Oracle and DB2 this is **Data Warehouse**. For target databases like Snowflake, Redshift, Greenplum, Netezza or PDW, the Data Warehouse connection can be renamed to **Repository.**
Connection Type	Indicates the connection source type or the connection method, such as Database, ODBC, Windows, Unix. Select the **Database** connection type.
Database Type	Type of database such as DB2, Greenplum, Hive, Netezza, Oracle, SQL Server, Teradata. Default is **(local)**. • If the database or metadata repository is located on the same server as the RED instance, leave the **(local)** default option selected. • If the database or metadata repository is located on a different server select the relevant database type of SQL Server, Oracle or DB2.
ODBC Data Source Name (DSN)	ODBC Data Source Name (DSN) as defined in the Windows 32-bit **ODBC Data Source Administrator**. • Select the relevant ODBC connection to the data warehouse database or metadata repository. **Note:** The ODBC Source Name defined in RED must be the same on all machines that use the corresponding connection.

© Copyright, WhereScape Software Ltd All rights reserved.

Options	Description
WhereScape RED Metadata Connection Indicator	Distinguishes the special connection that identifies the WhereScape RED data warehouse/metadata repository. This option must be enabled for **Data Warehouse/Metadata Repository** type connections. **Note:** There must only be one data warehouse/metadata connection in a WhereScape RED repository.

Source System

Options	Description
Database ID	Database Identifier (e.g. Oracle SID or TNS Name, Teradata TDPID) or Database Name (e.g. as in DB2 or SQL Server).
Database Link Name	Optional name of a Database Link that is used to access the database. Only required for Database type connections where the database is not on the same server as the data warehouse. If the server is on the same database, the link doesn't need to be defined and the field can be left blank. For connections to databases located in different servers where a database link load is required: • If the database link already exists to this source database, then enter that link name in this field. • If the link doesn't exist then enter a link name. Enter the defined SQL LINK name that points to the correct server, the link name does not need to be the same name as the server name.  Under some circumstances Oracle insists that this link name is the same as the tnsnames entry for the database, so it may be good practice to use the source SID as the database link name. If this is a new link then see the notes later in this section on **database link creation**).
Provider Name	Name of the connection provider/driver used to connect to the database.
Full Database Path Name	This field is used when you enter **Microsoft.Jet.OLEDB.4.0** as the Provider Name, e.g. when the source database is from MS Access or Excel. Enter the full database path name.

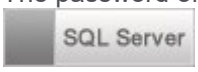
Big Data Adapter Settings

Options	Description
JDBC Connection String (JDBC URL)	Connection string used by the WhereScape Big Data Adapter to access this database.
JDBC Driver Class Name	JDBC driver class to be used by the WhereScape Big Data Adapter. This field must be set if the JDBC URL is set. • Select the appropriate JDBC Driver class name from the drop-down list. If this is left empty, this is not specified in generated commands.
Omit Sqoop Driver Option	• If set, the --driver option to Sqoop is omitted. This is required for certain connection types such as Oracle connections. • If you select the **Omit Sqoop Driver Option** check-box, the driver parameter is not used in Sqoop command line. This is a requirement for Oracle at the moment, as suggested by Sqoop documentation for 1.4.5.
Sqoop Connection Manager Class	Custom Sqoop connection manager class. Corresponds to the --connection-manager command line argument. Leave blank if this is not required.
Include Database/Schema	• If set, the --table option to Sqoop includes the database/schema name of the destination table when performing Apache Sqoop loads into this connection.

© Copyright, WhereScape Software Ltd All rights reserved.

Options	Description
Name in Sqoop Table Option	This is incompatible with some connection managers, for example the Cloudera Connector Powered by Teradata. • If this is not set, users must ensure that the database/schema is otherwise communicated to Sqoop, for example by using the $OBJECT_DATABASE$ token in the in the JDBC Connection String.
Include Sqoop Columns Option	• If set, the --columns option to Sqoop is included when performing Apache Sqoop loads into this connection. This is incompatible with some connection managers, for example the Cloudera Connector Powered by Teradata. • If this is not set, users must ensure that the order of columns in the loads match the order in the metadata.

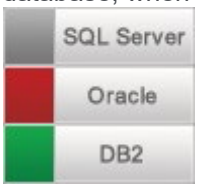
Database Credentials

Options	Description
Extract User ID	Database User that has access to SELECT from the source system tables to extract data. For SQL Server, this field can be left blank if using a trusted login, or the server login password. Refer to **Remove View Extract - Oracle Databases Only** for details.
Extract User Password	The password of the data warehouse user.  For SQL Server, this field can be left blank if using a trusted login, or the server login password.
Administrator User ID	Database User ID to login, when using WhereScape SQL Admin via the 'Connect using SQL Admin (as SQL Admin User)' context menu item (optional).
Administrator User Password	Optional Database User Password to login, when using WhereScape SQL Admin via the 'Connect using SQL Admin (as SQL Admin User)' context menu item.
JDBC User ID	User ID to login as using JDBC via WhereScape Big Data Adapter (optional).
JDBC Password	Password to login with when using JDBC via WhereScape Big Data Adapter (optional).

Other

Options	Description
Default Schema for Browsing	Optional comma-delimited list of schemas for the browser pane filter. Enter the schema(s) you want the connection to browse by default on the right browser pane.
New Table Default Load Type	The default **Load Type** for new Load tables created using this connection as a source. Select the desired default load type from the list, e.g. **Database Link Load, Script based load, Integration Services Load** or **Externally Loaded**. **Note:** The available options in this drop-down list is configured from Tools > Options > **Available Load Types**.
New Table Default Load Script Connection	The default **Script Connection** to use when a **Script based load** type is defined for a Load table object that is **sourced** from this connection.
New Table Default Load Script Template	The default **Script Template** to use when a **Script based load** type is defined for a Load table object that is **sourced** from this connection.
SSIS Connection String (OLEDB)	Connection string to be used by Microsoft SQL Server Integration Services (SSIS) to connect to the data source or destination. For details on how to use the wizard to build an SSIS Connection String, refer to **Loading Data into RED Load Tables using SSIS** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

Options	Description
	Note: A connection string is typically composed of multiple property name/value pairs that are semi-colon delimited.
Data Type Mapping Set	Mapping Set to use when converting from a source database data type to a destination database data type. Setting this field to **(Default)** makes RED automatically select the relevant mapping set, otherwise you can choose one of the standard mapping sets from the drop-down list or create a new set.
Default Transform Function Set	Function Set that is selected by default in the Transformation dialogs.
When Connection is an OLAP Data Source	This section of fields is only relevant and will only be visible from a **Data warehouse** connection (where the Data warehouse field is enabled). These fields are required so that the data warehouse can be used as a source for the Analysis Services cubes.
MSAS Connection String	Connection string to be used by Microsoft Analysis Services (MSAS) to connect to the data warehouse. For details on how to use the wizard to build the OLAP connection string, refer to **OLAP Defining Data Source for the OLAP Cube**. **Note:** A connection string is typically composed of multiple property name/value pairs that are semi-colon delimited.
Connection Provider/Driver	Name of the Connection Provider/Driver to use to connect to the data warehouse database, when it is used as the data source for OLAP cubes.  set to **SQLOLEDB**. set to **MSDAORA**. set to **IBMDADB2**.
Data Warehouse Server	Data Warehouse Server Name, which is used when the data warehouse is used as the data source for OLAP cubes.
Data Warehouse Database ID	Data Warehouse Database Identifier (e.g. Oracle SID or TNS Name, Teradata TDPID) or Database Name (e.g. as in DB2 or SQL Server), which is used when the data warehouse is used as the data source for OLAP cubes.

Tip

Once the connection has been set up, you can right-click the connection in the middle pane or double click the connection name from the left pane to view or edit the connection's **Properties**.

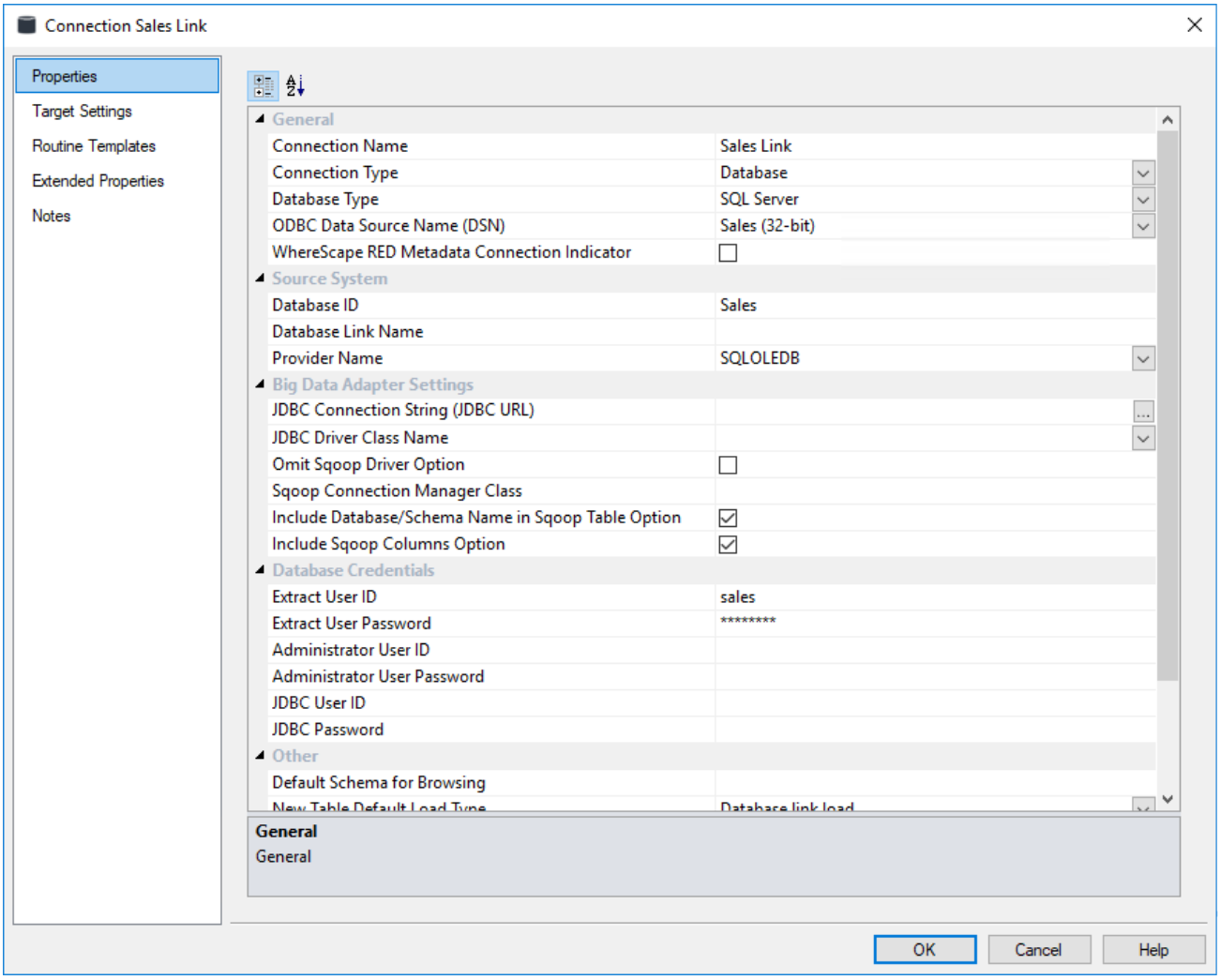
© Copyright, WhereScape Software Ltd All rights reserved.

Database

This topic describes the details of the connection Properties as they apply to the Database type connections.

When coming to the Data Warehouse from an OLTP source system, the connection defines the path to get to those source tables. A connection object must be defined for each source system.

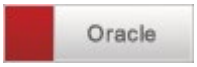
The fields presented in the **Connection Properties** screen below, depends on the connection and database types selected. See below for a description of all possible fields.



SQL Server: when running a SQL Server data warehouse it is possible to create database links (linked server) to other SQL Server instances and most of the other major databases, such as Oracle, DB2, etc.

- Database link load - from a table in the current SQL Server database.
- Database link load - from a table in another database on the same SQL Server instance - the OBDC Data Source Name must be defined in the Windows 32-bit **ODBC Data Source Administrator** and selected on the ODBC DSN field, as well as the Database ID that also needs to be populated in the Database ID field in the connection properties.
- Database link load - through a linked server.
- Integration Services load.

© Copyright, WhereScape Software Ltd All rights reserved.



Oracle: using the Oracle database it is possible to create a database link to another Oracle database as standard, or to any other database by using Oracles Heterogeneous Services. Refer to Oracle Examples.

- Database link load - from a table in the current Oracle database server (includes Oracle gateways and Oracle heterogeneous services).
- Database link load - through a database link to another Oracle server.

DB2: using the DB2 database it is possible to create a database link load to another table in the same DB2 system or to another system via CLI. Refer to DB2 Examples.

- Database link load - from another table in the same DB2 system (includes federation).
- Database link load - via CLI from a DB2 table in another system.

Examples for each of RED's database type connection screens for SQL Server, Oracle, DB2 Greenplum, Netezza and PDW are provided below the connection properties description.

A Database Link Load through a **linked server** can have a number of source databases:

- SQL Server
- Oracle
- DB2 for Linux, UNIX and Windows (the DB2 RED can build a warehouse in)
- Teradata
- Greenplum
- Netezza
- PDW
- Hive
- Custom
- Sybase
- DB2/400 (the DB2 that runs on AS/400 machines - quite different to above)
- MS Access
- Excel

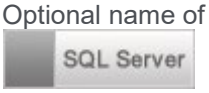
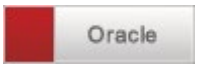
The connection object Properties window has the following fields:

General

Options	Description
Connection Name	Name used to label the connection within WhereScape RED.
Connection Type	Indicates the connection source type or the connection method, such as Database, ODBC, Windows, Unix. Select the **Database** connection type.
Database Type	Type of database such as DB2, Greenplum, Hive, Netezza, Oracle, SQL Server, Teradata. Default is **(local)**.
ODBC Data Source Name (DSN)	ODBC Data Source Name (DSN) as defined in the Windows 32-bit **ODBC Data Source Administrator**. **Note:** The ODBC Source Name defined in RED must be the same on all machines that use the corresponding connection.
WhereScape RED Metadata Connection Indicator	Only required for the Data Warehouse/metadata repository connection. Leave this check box unselected.

© Copyright, WhereScape Software Ltd All rights reserved.

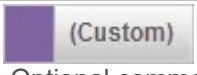
Source System

Options	Description
Provider Name	Name of the connection provider/driver used to connect to the database.
Database ID	Database Identifier (e.g. Oracle SID or TNS Name, Teradata TDPID) or Database Name (e.g. as in DB2 or SQL Server).
Database Host/Server	**PDW Database type connections only:** Used for sqlcmd specification.
Database Port	**PDW Database type connections only:** Used for sqlcmd specification.
Database Link Name	Optional name of a Database Link that is used to access the database.  If SQL Server and the database are on the same server as the data warehouse then no link needs to be defined and this field can be left blank.  If a database link already exists to this source database then enter that link name in this field. If no link exists then enter a link name. Under some circumstances Oracle insists that this link name is the same as the tnsnames entry for the database, so it may be good practice to use the source SID as the database link name. If this is a new link then see the notes later in this section on **Creating a Database Link**. Enter the defined SQL LINK name that points to the correct server, the link name does not need to be the same name as the server name.
Provider Name	Name of the connection provider/driver used to connect to the database.
Full Database Path Name	This field is used when you enter **Microsoft.Jet.OLEDB.4.0** as the Provider Name, e.g. when the source database is from MS Access or Excel. Enter the full database path name.

Database Credentials

Options	Description
Extract User ID	Database User that has access to SELECT from the source system tables to extract data. For more information, refer to **Remote View Extract - Oracle Databases only**.
Extract User Password	The password of the data warehouse user. For SQL Server blank if a trusted login, or the server login password.
Teradata Wallet User ID/Teradata Wallet String	If connecting to a **Teradata** server, the Teradata Wallet log on method can be selected from the ODBC User Default drop-down menu and the Teradata Wallet User ID and String can be entered, instead of User/Password.
Administrator User ID	Database User ID to login, when using WhereScape SQL Admin via the 'Connect using SQL Admin (as SQL Admin User)' context menu item (optional).
Administrator User Password	Optional Database User Password to login, when using WhereScape SQL Admin via the 'Connect using SQL Admin (as SQL Admin User)' context menu item.

Other

Options	Description
Default Database for Browsing	 Optional comma-delimited list of databases for the browser pane filter. Enter the database(s) you want the connection to browse by default on the right browser pane.
Default Schema for Browsing	Optional comma-delimited list of schemas for the browser pane filter. Enter the schema(s) you want the connection to browse by default on the right browser pane.
New Table Default Load Type	The default **Load Type** for new Load tables created using this connection as a source. Select the desired default load type from the list, e.g. **Database Link Load,**

© Copyright, WhereScape Software Ltd All rights reserved.

Options	Description
	Script based load, Integration Services Load or **Externally Loaded**. **Note:** The available options in this drop-down list is configured from Tools > Options > **Available Load Types**.
New Table Default Load Script Connection	The default **Script Connection** to use when a **Script based load** type is defined for a Load table object that is **sourced** from this connection.
New Table Default Load Script Template	The default **Script Template** to use when a **Script based load** type is defined for a Load table object that is **sourced** from this connection.
SSIS Connection String (OLEDB)	Connection string to be used by Microsoft SQL Server Integration Services (SSIS) to connect to the data source or destination. For details on how to use the wizard to build an SSIS Connection String, refer to **Loading Data into RED Load Tables using SSIS** for details. **Note:** A connection string is typically composed of multiple property name/value pairs that are semi-colon delimited.
SSIS Connection String (SQLPDW)	 **PDW Database type connections only:** Connection string to be used by Microsoft SQL Integration Services (SSIS) to connect to the data source or destination using the SQL Server Parallel Data Warehouse Connection Manager. **Note:** A connection string is typically composed of multiple property name/value pairs that are semi-colon delimited.
Staging database	 **PDW Database type connections only:** The staging database to be used by Microsoft SQL Server Integration Services (SSIS) for loading to PDW, if this option is enabled in the **Flat File Load - Source screen**'s Properties.
Data Type Mapping Set	Mapping Set to use when converting from a source database data type to a destination database data type. Setting this field to **(Default)** makes RED automatically select the relevant mapping set, otherwise you can choose one of the standard mapping sets from the drop-down list or create a new set.

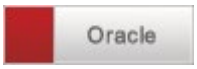
Tip:

Once the connection has been set up, you can right-click the connection in the middle pane or double click the connection name from the left pane to view or edit the connection's **Properties**.

© Copyright, WhereScape Software Ltd All rights reserved.

ODBC

This connection type is via an ODBC link. Most data movement is performed using the ODBC connection.



Oracle can perform **ODBC loads** using **Direct Path OCI** and **standard (single row inserts)** via the scheduler.

ODBC loads can have a number of source databases:

- SQL Server
- Oracle
- DB2 for Linux, UNIX and Windows (the DB2 RED can build a warehouse in)
- Teradata
- Greenplum
- Netezza
- PDW
- Hive
- Sybase
- MySQL
- DB2/400 (the DB2 that runs on AS/400 machines - quite different to above)
- MS Access
- Excel

The connection object Properties window has the following fields:

General

Options	Description
Connection Name	Name used to label the connection within WhereScape RED. Typically for SQL Server, Oracle and DB2 this is **Data Warehouse**. For target databases like Greenplum, Netezza or PDW the Data Warehouse connection can be renamed to **Repository.**
Connection Type	Indicates the connection source type or the connection method, such as Database, ODBC, Windows, Unix. Select the **ODBC** connection type.
Database Type	Type of database such as DB2, Greenplum, Hive, Netezza, Oracle, SQL Server, Teradata. Default is **(local)**.
ODBC Data Source Name (DSN)	ODBC Data Source Name (DSN) as defined in the Windows 32-bit **ODBC Data Source Administrator**. **Note:** The ODBC Source Name defined in RED must be the same on all machines that use the corresponding connection.
WhereScape RED Metadata Connection Indicator	Distinguishes the special connection that identifies the WhereScape RED data warehouse/metadata repository. This option must be enabled for **Data Warehouse/Metadata Repository** type connections. **Note:** There must only be one data warehouse/metadata connection in a WhereScape RED repository.

ODBC

Options	Description
Database ID	Source Database Identifier (e.g. Oracle SID or TNS Name, Teradata TDPID) or Database Name (e.g. as in DB2 or SQL Server).
Work Directory	Windows directory used by WhereScape RED to create temporary files for minimal logged extracts. The directory must exist and allow write access. There must be a different work directory for each WhereScape RED Scheduler running

© Copyright, WhereScape Software Ltd All rights reserved.

Options	Description
	on the same machine to avoid file conflicts. Typically **C:\Temp** or a **sub-directory of C:\Temp** is used.

Credentials

Note:

If the source database does not support windows authentication, a username and password must be specified in the User ID and User Password fields below.

Options	Description
Extract User ID	Database User that has access to SELECT from the source system tables to extract data.
Extract User Password	Database Password to use with the Extract User ID to login to the source system to extract data.
Administrator User ID	Leave blank.
Administrator User Password	Leave blank.

Other

Options	Description
Default Schema for Browsing	Optional comma-delimited list of schemas for the browser pane filter. Enter the schema(s) you want the connection to browse by default on the right browser pane.
New Table Default Load Type	The default **Load Type** for new Load tables created using this connection. Select the desired default load type from **Native ODBC, Integration Services load** or **ODBC load**. **Note:** The available options in this drop-down list is configured from Tools > Options > **Available Load Types**.
New Table Default Load Script Connection	The default **Script Connection** to use when a **Script based load** type is defined for a Load table object that is **sourced** from this connection.
New Table Default Load Script Template	The default **Script Template** to use when a **Script based load** type is defined for a Load table object that is **sourced** from this connection.
SSIS Connection String (OLEDB)	Connection string to be used by Microsoft SQL Server Integration Services (SSIS) to connect to the data source or destination. For details on how to use the wizard to build an SSIS Connection String, refer to **Loading Data into RED Load Tables using SSIS** for details. **Note:** A connection string is typically composed of multiple property name/value pairs that are semi-colon delimited.
SSIS Connection String (SQLPDW)	**PDW Database type connections only:** Connection string to be used by Microsoft SQL Integration Services (SSIS) to connect to the data source or destination using the SQL Server Parallel Data Warehouse Connection Manager. The **SSIS Connection String (SQLPDW)** is used when the target is PDW and it must be set in order to process PDW loads. **Note:** A connection string is typically composed of multiple property name/value pairs that are semi-colon delimited.

© Copyright, WhereScape Software Ltd All rights reserved.

Options	Description
Staging database	**PDW Database type connections only:** The staging database to be used by Microsoft SQL Server Integration Services (SSIS) for loading to PDW.
Data Type Mapping Set	Mapping Set to use when converting from a source database data type to a destination database data type. Setting this field to **(Default)** makes RED automatically select the relevant mapping set, otherwise you can choose one of the standard mapping sets from the drop-down list or create a new set.
When Connection is an OLAP Data Source	This section of fields is only relevant and will only be visible from a **Data warehouse** connection (where the Data warehouse field is enabled). These fields are required so that the data warehouse can be used as a source for the Analysis Services cubes.
MSAS Connection String	Connection string to be used by Microsoft Analysis Services (MSAS) to connect to the data warehouse. For details on how to use the wizard to build the OLAP connection string, refer to **OLAP Defining Data Source for the OLAP Cube**. **Note:** A connection string is typically composed of multiple property name/value pairs that are semi-colon delimited.
Connection Provider/Driver	Name of the Connection Provider/Driver to use to connect to the data warehouse database, when it is used as the data source for OLAP cubes. set to **SQLOLEDB**. set to **MSDAORA**. set to **IBMDADB2**.
Data Warehouse Server	Data Warehouse Server Name, which is used when the data warehouse is used as the data source for OLAP cubes.
Data Warehouse Database ID	Data Warehouse Database Identifier (e.g. Oracle SID or TNS Name, Teradata TDPID) or Database Name (e.g. as in DB2 or SQL Server), which is used when the data warehouse is used as the data source for OLAP cubes.

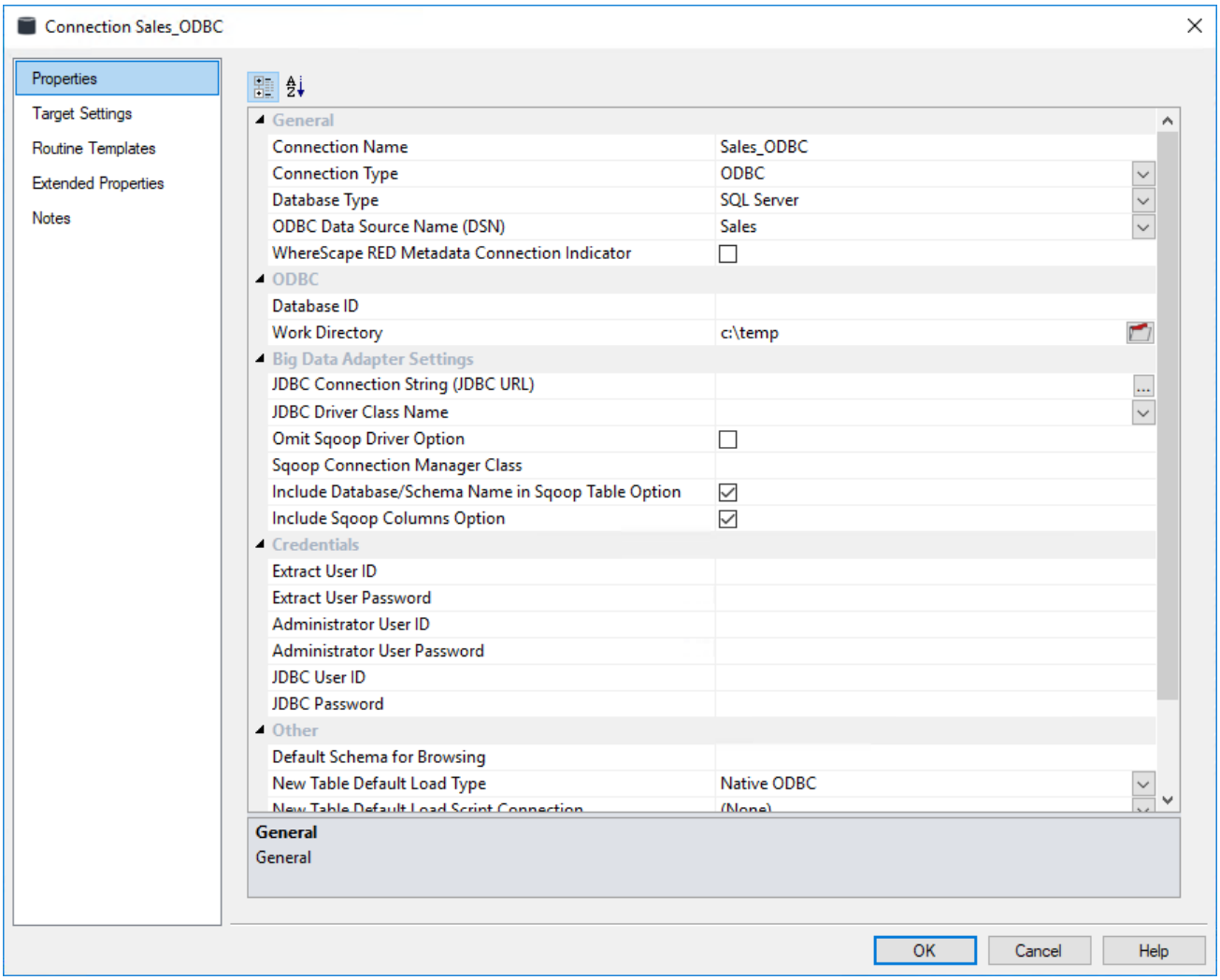
Tip:

Once the connection has been set up, you can right-click the connection in the middle pane or double click the connection name from the left pane to view or edit the connection's **Properties**.

© Copyright, WhereScape Software Ltd All rights reserved.

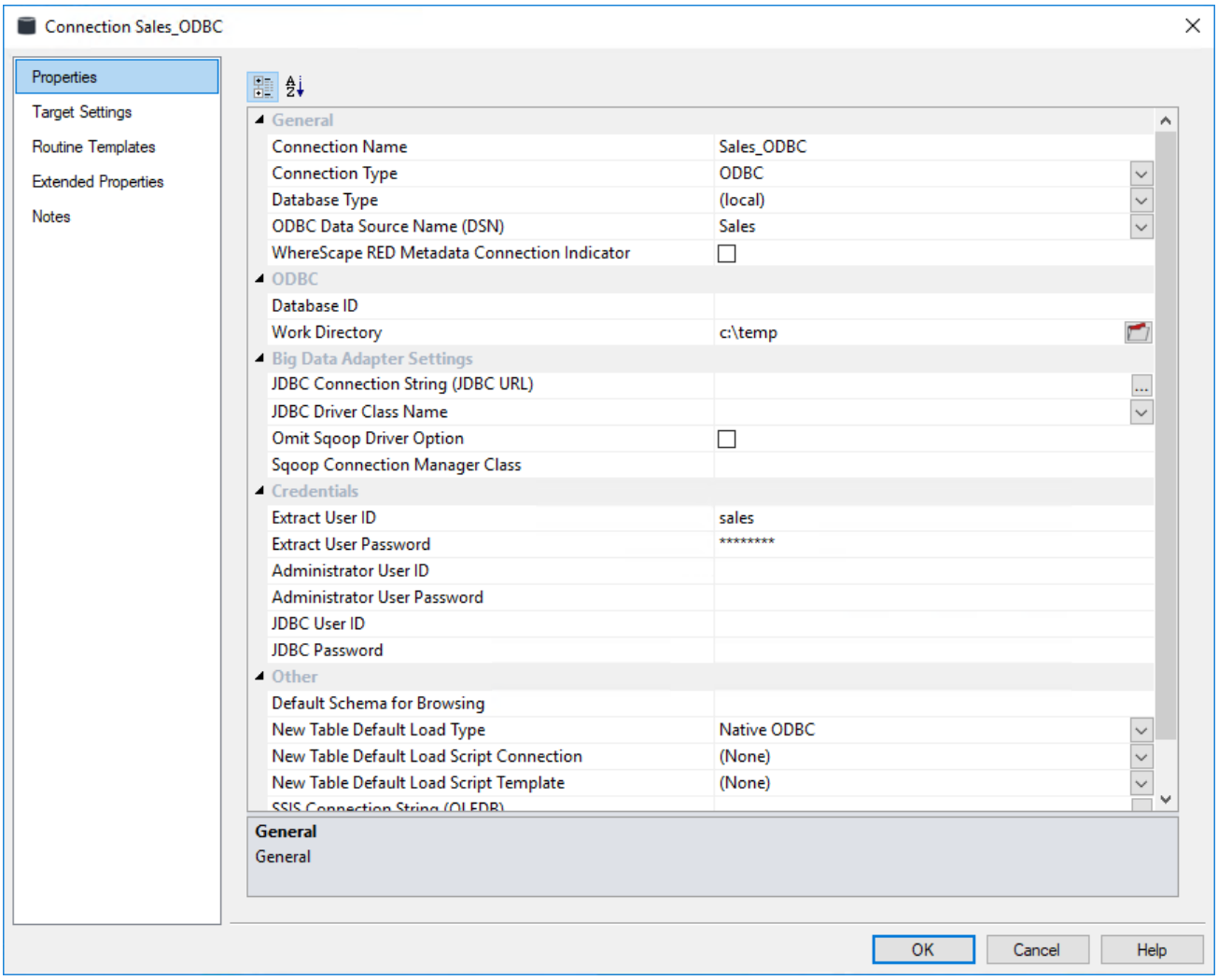
Below are two examples of a **Native ODBC** Load.

If the source database supports windows authentication, it is not necessary to specify a username and password:



© Copyright, WhereScape Software Ltd All rights reserved.

If the source database does not support windows authentication, a username and password must be specified:

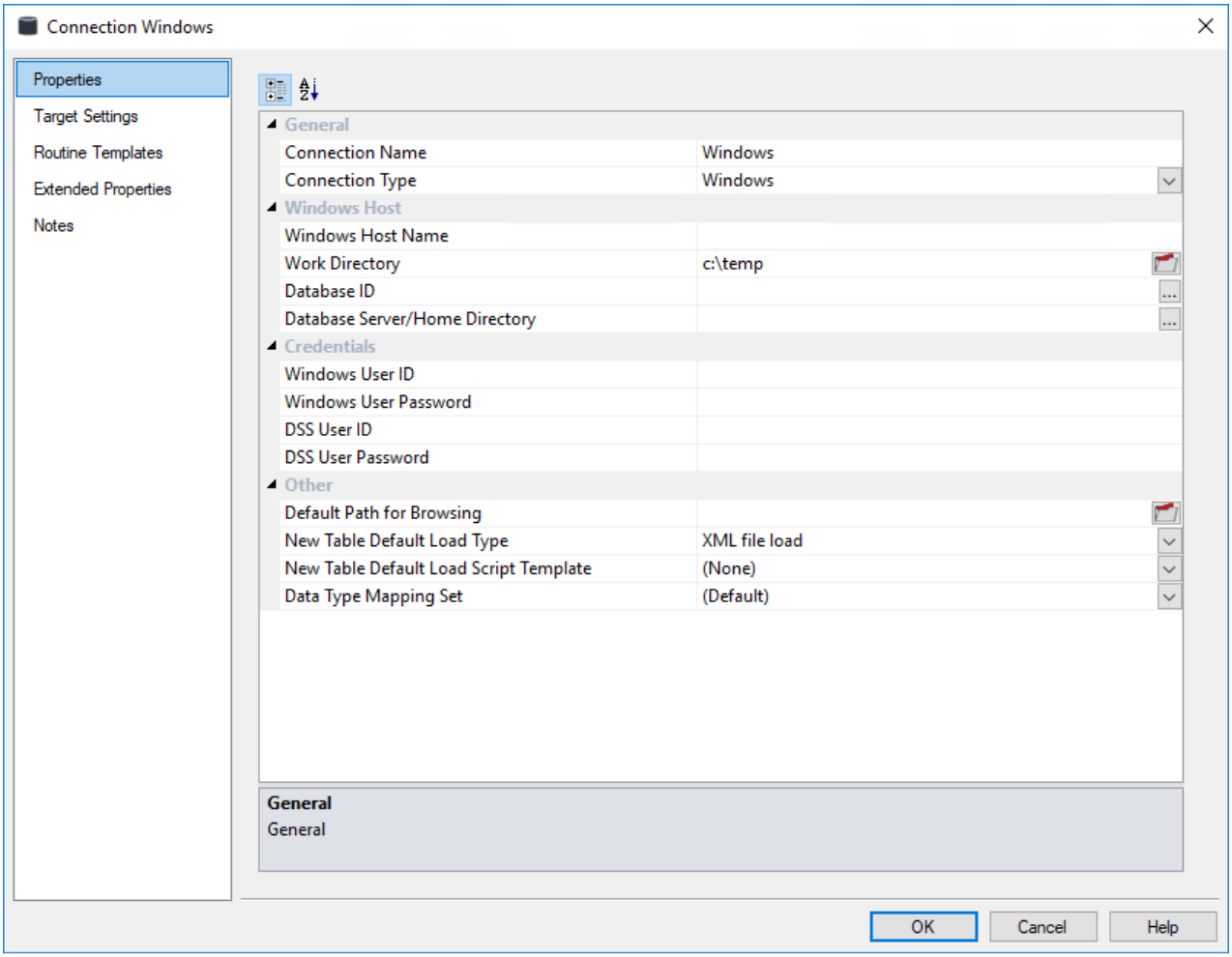


© Copyright, WhereScape Software Ltd All rights reserved.

Windows

This connection is for the PC that you are working on, or to a host Windows PC. From Windows, you can only do flat file loads or file loads using SSIS.
Refer to **Loading Data into RED Load Tables using SSIS** for details about SSIS File Loads.

Sample **SQL Server Windows Connection** screen:



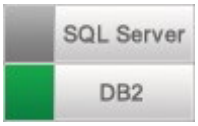
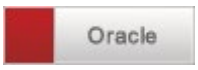
General

Options	Description
Connection Name	Name used to label the connection within WhereScape RED.
Connection Type	Indicates the connection source type or the connection method, such as Database, ODBC, Windows, Unix. Select the **Windows** connection type.

Windows Host

Options	Description
Windows Host Name	IP address or host name that identifies the Windows machine. Leave blank to connect to the local machine.
Work Directory	Windows directory used by WhereScape RED to create temporary files for minimal logged extracts. The directory must exist and allow write access. There must be a different work directory for each WhereScape RED Scheduler running on the same machine to avoid file conflicts. Typically C:\Temp or a sub-directory of C:\Temp is used.

© Copyright, WhereScape Software Ltd All rights reserved.

Options	Description
Database ID	The Source Database Identifier.  The **database name** of the Data Warehouse.  The Oracle **SID** or **TNS Name**.
Database Server/Home Directory	Optional to specify the Database Home Directory if it is different from the standard home directory. The server name (or server/instance name) for the Data Warehouse database.

Credentials

Options	Description
Windows User ID and Password	Leave this blank if you are connecting to your own PC. Enter details if you are connecting remotely to another Windows system.
Dss User ID and Password	Enter the relevant details for connecting to the **Data Warehouse**.

Other

Options	Description
Default Path for Browsing	Optional default Path for browser pane filter. When a path has been selected in this field, it becomes the initial point for browsing and it is also expanded on open in the right hand browser pane.
New Table Default Load Type	The default **Load Type** for new Load tables created using this connection as a source. Select the desired default load type from the list, e.g. **Database Link Load, Script based load, Integration Services Load** or **Externally Loaded**. **Note:** The available options in this drop-down list is configured from Tools > Options > **Available Load Types**.
New Table Default Load Script Template	The default **Script Template** to use when a **Script based load** type is defined for a Load table object that is **sourced** from this connection.
Data Type Mapping Set	Mapping Set to use when converting from a source database data type to a destination database data type. Setting this field to **(Default)** makes RED automatically select the relevant mapping set, otherwise you can choose one of the standard mapping sets from the drop-down list or create a new set.

Tip:

Once the connection has been set up, you can right-click the connection in the middle pane or double click the connection name from the left pane to view or edit the connection's **Properties**.

To test the connection:

- Select **Browse** | **Source Tables** from the menu strip
- In the right pane you should be able to drill down to the area required.

© Copyright, WhereScape Software Ltd All rights reserved.

UNIX

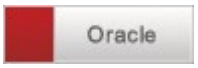
Tip:

UNIX/Linux connections are only supported from **Oracle,DB2** and **Greenplum** data warehouses.

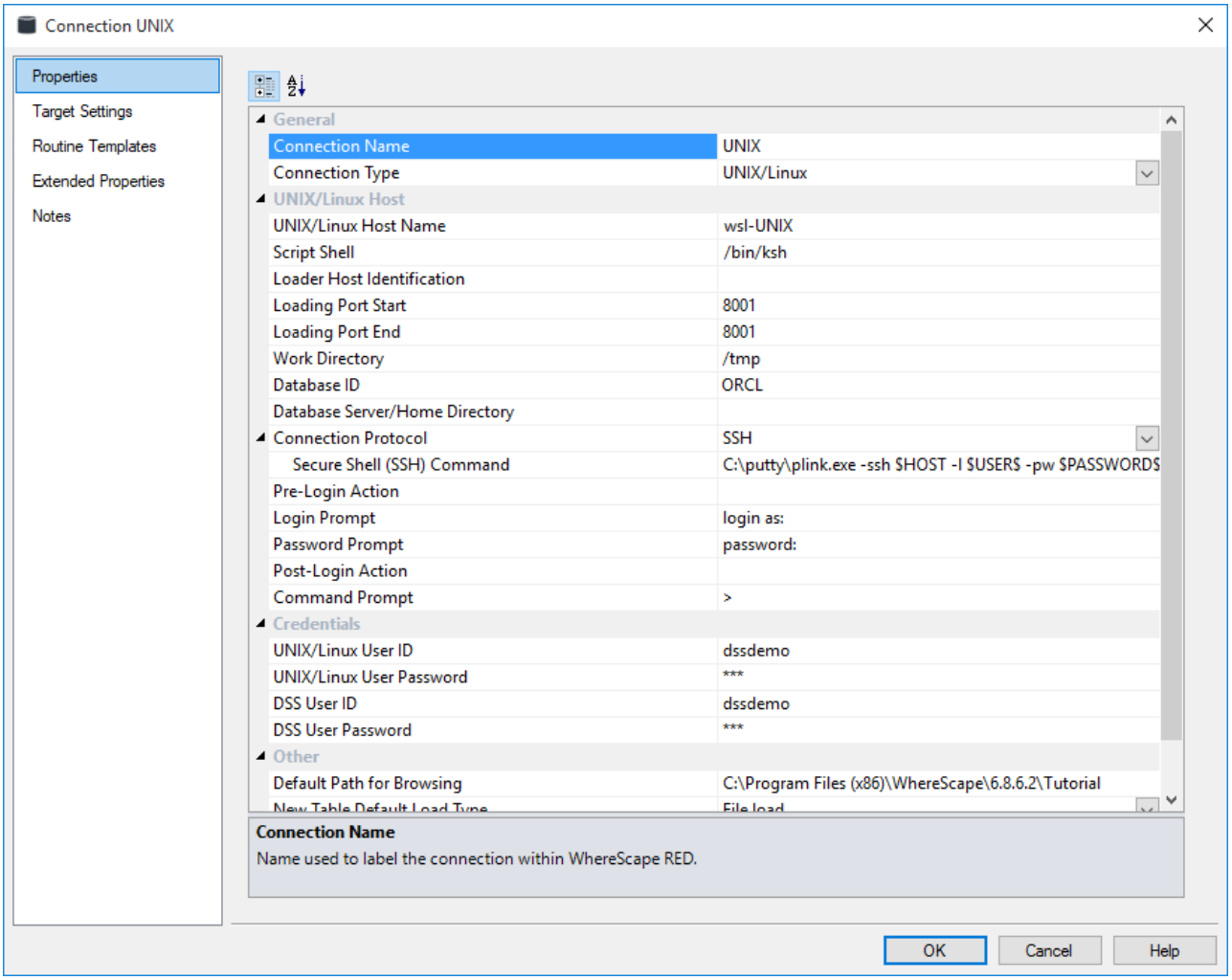
If the **UNIX/Linux** connection returns a blank screen or an error message in the **Results** pane after the connection is browsed, take necessary action through the **Server (SSH)** tab next to the main Builder and Scheduler tabs.

This tab is displayed after browsing the UNIX connection.

This topic describes in greater detail the connection Properties as they apply to **UNIX** connections. From a UNIX connection you can only do flat file loads.

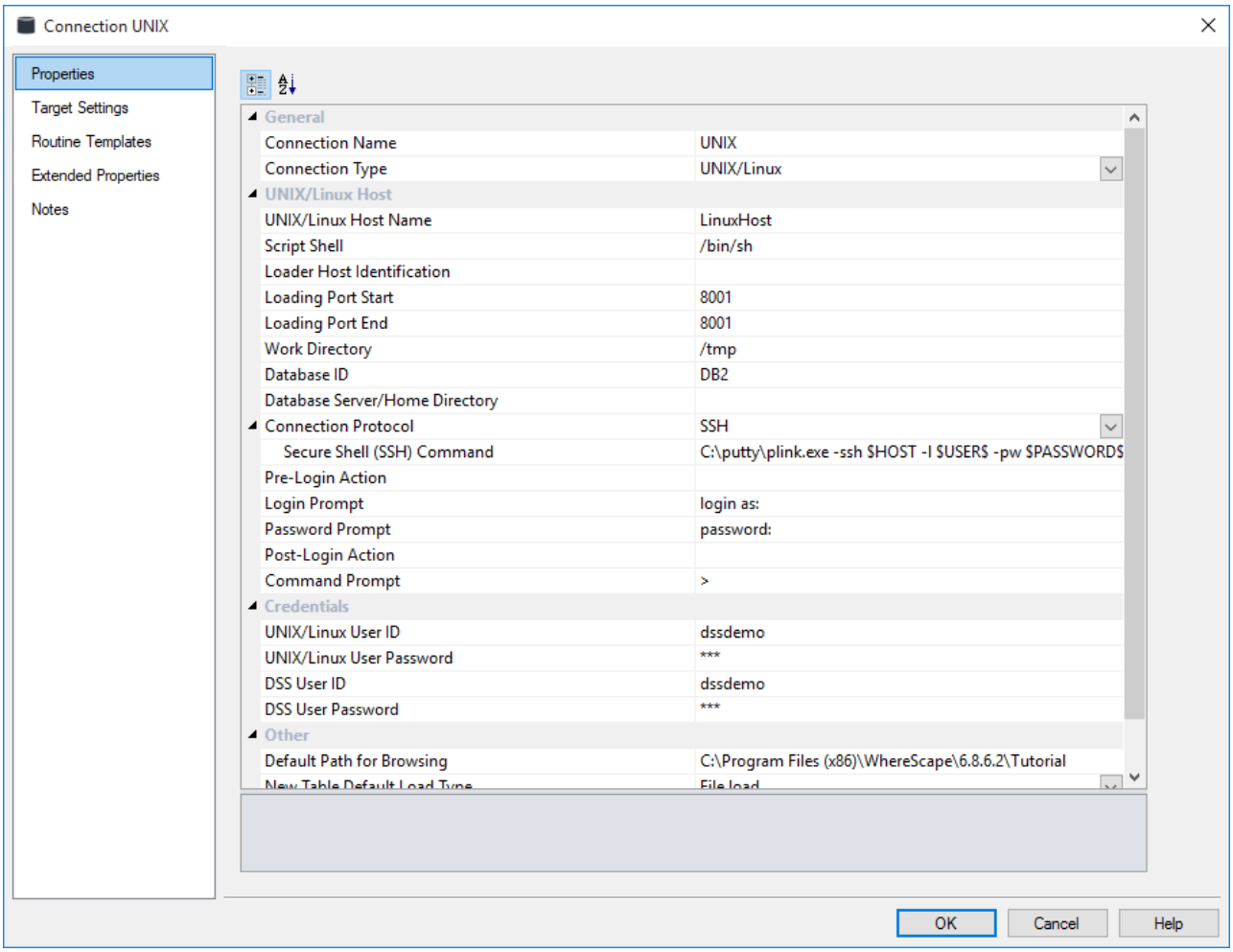


Sample Oracle UNIX connection screen



© Copyright, WhereScape Software Ltd All rights reserved.

Sample DB2 UNIX connection screen



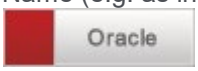
General

Options	Description
Connection Name	Name used to label the connection within WhereScape RED.
Connection Type	Indicates the connection source type or the connection method, such as Database, ODBC, Windows, Unix. Select the **UNIX** connection type.

Unix/Linux Host

Options	Description
UNIX/Linux Host Name	IP address or host name that identifies the UNIX machine.
Script Shell	Path to the POSIX-compliant UNIX/Linux shell to use for generated scripts. For **UNIX** hosts, set to **/bin/ksh**. For **Linux** hosts set to **/bin/sh**. If this field is left blank, a default will be chosen based on the name of the connection and the type of database used for the WhereScape RED metadata repository.
Loader Host Identification	IP Address or host name(s) that identifies the Loader/ Multiple hosts can be entered with using a comma (,) to delimit.
Work Directory	Windows directory used by WhereScape RED to create temporary files for minimal logged extracts. The directory must exist and allow write access. There must be a different work directory for each WhereScape RED Scheduler running on the same machine to avoid file conflicts. Typically C:\Temp or a sub-directory of C:\Temp is used.

© Copyright, WhereScape Software Ltd All rights reserved.

Options	Description
Database ID	Database Identifier (e.g. Oracle SID or TNS Name, Teradata TDPID) or Database Name (e.g. as in DB2 or SQL Server).  For UNIX/Linux exports and loads, if the ORACLE_SID environment variable is set in your UNIX/Linux environment, this will be the variable used. If this environment variable is not set then the value from the UNIX/Linux connection object will be used.
Database Server/Home Directory	Optional to specify the Database Home Directory if it is different from the standard home directory.
Connection Protocol	Telnet or Secure Shell (SSH) protocol to use to connect to the UNIX/Linux machine. For SSH, the **Secure Shell (SSH) Command** property is enabled to specify how to connect.
Secure Shell (SSH) Command	Command to execute to connect to a UNIX/Linux machine using the Secure Shell (SSH) protocol such as **C:\Program Files(x86)\PuTTY\plink.exe -ssh $HOST$ -l $USER$ -pw $PASSWORD$**
Pre-Login Action, Login Prompt, Password Prompt, Post-Login Action, and Command Prompt.	These fields are only used to create a Telnet connection to the host machine. WhereScape RED uses the Telnet connection in the drag and drop functionality. They are **not used in the actual production running of the Data Warehouse,** and is only necessary if you wish to use the drag and drop functionality.
Pre-Login Action	Response or command to send BEFORE logging in to the UNIX/Linux machine. Typically this is NOT necessary but it can be used to indicate that the UNIX/Linux Login Prompt is preceded by a line-feed (\n). However it is preferable that the UNIX/Linux login displays the Login Prompt without anything preceding it. [Optional]
Login Prompt	The UNIX login prompt, or the tail end of the login prompt, e.g. **ogin as:**.
Password Prompt	The UNIX password prompt, or the tail end of the password prompt, e.g. **ssword:**.
Post-Login Action	Not often used but may be necessary to respond to a login question. It is preferable that the UNIX login goes straight to the command prompt.
Command Prompt	Enter the UNIX/Linux command prompt, or the tail end of that prompt, typically **>**. **Note** To ascertain some of the above fields, you have to log in to the UNIX system.

Credentials

Options	Description
UNIX/Linux User ID	User Account to login to the UNIX/Linux Host.
UNIX/Linux User Password	Password to login to the UNIX/Linux Host.
DSS User ID	Database user to connect to the WhereScape RED metadata repository.
DSS User Password	Database password to connect to the WhereScape RED metadata repository.

Other

Options	Description
Default Path for Browsing	Optional default Path for browser pane filter. When a path has been selected in this field, it becomes the initial point for browsing and it is also expanded on open in the right hand browser pane.
New Table Default Load Type	The default **Load Type** for new Load tables created using this connection as a source. Select the desired default load type from the list, e.g. **Database Link Load,**

© Copyright, WhereScape Software Ltd All rights reserved.

Options	Description
	Script based load, Integration Services Load or **Externally Loaded**. **Note:** The available options in this drop-down list is configured from Tools > Options > **Available Load Types**.
New Table Default Load Script Template	The default **Script Template** to use when a **Script based load** type is defined for a Load table object that is **sourced** from this connection.
Data Type Mapping Set	XML files have been created to store mappings from one set of data types to another. Setting this field to **(Default)** will cause RED to automatically select the relevant mapping set; otherwise you can choose one of the standard mapping sets from the drop-down list or create a new one.

To validate the fields

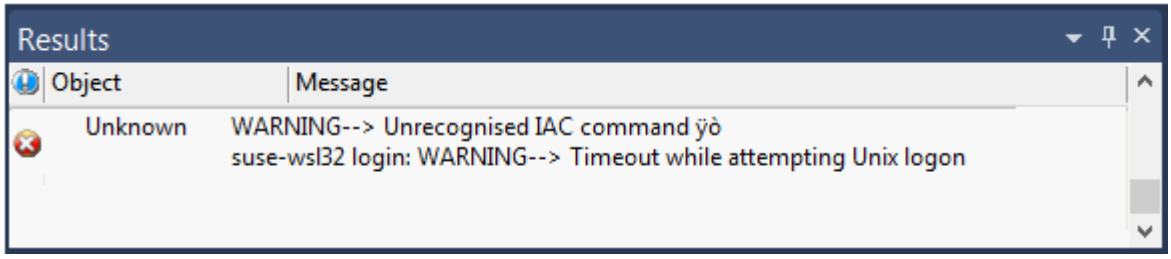
- Right-click on the connection name
- Select **Telnet window**

To test the drag and drop functionality

- From the menu strip select **Browse > Source Tables**
- Drill down to the area required
- Drag an item to the middle pane, (having first selected the object in the left pane).

Connection Failures

In the event that a telnet connection cannot be established to the UNIX host, the following message normally appears in the results pane, after approximately 30 seconds



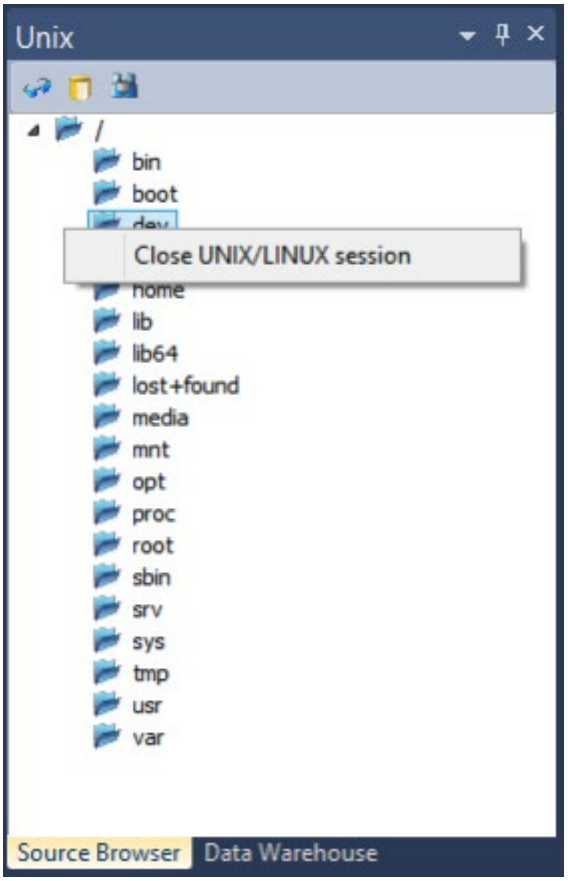
Attempt the connection again and using the **Window** menu option select the **Telnet** window. This displays the login session and should provide an insight as to why the connection is not being completed.

If the situation cannot be resolved, a telnet trace can be acquired. Select the **Tools > Options** menu option and click on the **Trace all Unix sessions** check box. Then try to do the connection or browse again. A log file called WslMedTelnet.log is created in the WhereScape program directory. Edit the log file and ensure there are no passwords visible and then contact WhereScape support using the WhereScape forum at www.wherescape.com.

© Copyright, WhereScape Software Ltd All rights reserved.

Closing the Connection

To close the collection, right-click in the browser pane and select **Close UNIX/LINUX session**:



© Copyright, WhereScape Software Ltd All rights reserved.

Hadoop

This topic describes in greater detail the connection properties, as they apply to **Hadoop** connections.

Hadoop as a source works only with connections to Hadoop from which users can only do flat file loads. The connection must be set via a Secure Shell (SSH) protocol.

For Hadoop script loads from Oracle using **Oracle's Big Data Connectors**, refer to **Hadoop using Oracle's Big Data Connectors** for details.

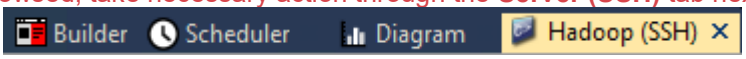
Note

WhereScape RED only fully supports HDFS as the underlying file system.

Tip

When the **Big Data Adapter** Settings are populated in Hadoop connections, RED can load data from Hadoop into Hive and/or Datawarehouse tables and also perform loads from Hadoop directly into the Datawarehouse using Sqoop through WhereScape RED's Big Data Adapter (BDA).
For more information about these settings, refer to the **Big Data Adapter Settings** fields description below and also refer to **Connections to the Data Warehouse/Metadata Repository**, **Configuring your database for use by BDA** and **Apache Sqoop Load**.

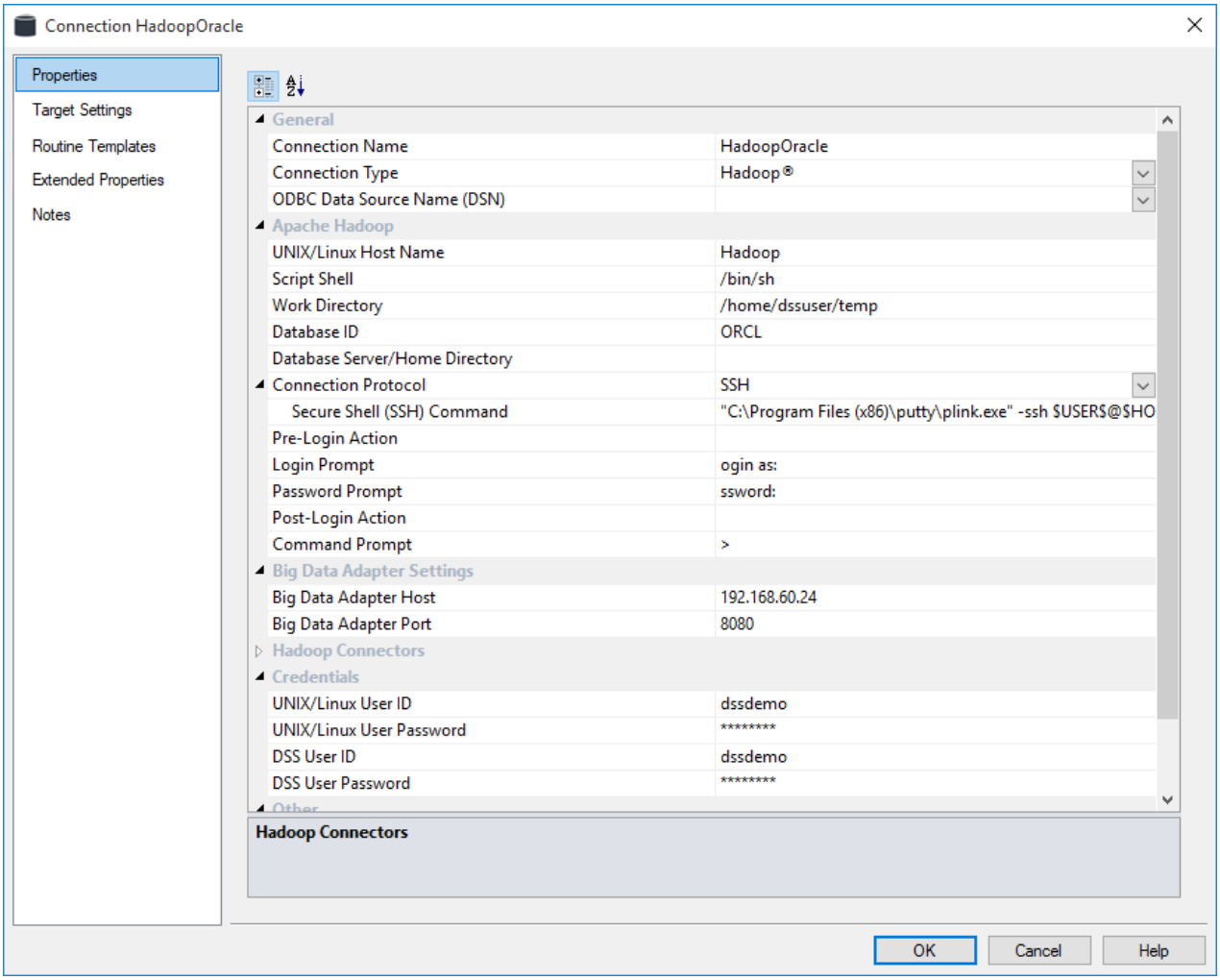
Note

If the **Hadoop** connection returns a blank screen or an error message in the **Results** pane after the connection is browsed, take necessary action through the **Server (SSH)** tab next to the main Builder and Scheduler tabs. 

This tab is displayed after browsing the UNIX connection.

© Copyright, WhereScape Software Ltd All rights reserved.

Sample Hadoop Connection screen:



General

Options	Description
Connection Name	Name used to label the connection within WhereScape RED.
Connection Type	Indicates the connection source type or the connection method, such as Database, ODBC, Windows, Unix. Select the **Hadoop** connection type.

Apache Hadoop

Options	Description
UNIX/Linux Host Name	IP address or host name that identifies the Hadoop server.
Script Shell	Path to the POSIX-compliant UNIX/Linux shell to use for generated scripts. For **UNIX** hosts, set to **/bin/ksh**. For **Linux** hosts set to **/bin/sh**. If this field is left blank, a default will be chosen based on the name of the connection and the type of database used for the WhereScape RED metadata repository.
Work Directory	Directory used by WhereScape RED to create temporary files for minimal logged extracts. The directory must exist and allow write access. There must be a different work directory for each WhereScape RED Scheduler running on the same machine to avoid file conflicts. Typically /tmp or a sub-directory of the UNIX user is used.
Database ID	Database Identifier (e.g. Oracle SID or TNS Name, Teradata TDPID) or Database Name (e.g. as in DB2 or SQL Server).
Database	Optional to specify the Database Home Directory if it is different from the standard

© Copyright, WhereScape Software Ltd All rights reserved.

Options	Description
Server/Home Directory	home directory.
Connection Protocol	Telnet or Secure Shell (SSH) protocol to use to connect to the Hadoop machine. For SSH, the **Secure Shell (SSH) Command** property is enabled to specify how to connect.
Secure Shell (SSH) Command	Command to execute to connect to a Hadoop machine using the Secure Shell (SSH) protocol, such as **C:\putty\plink.exe -ssh some_host_name**.
Pre-Login Action, Login Prompt, Password Prompt, Post-Login Action, and Command Prompt.	These fields are only used to create a Telnet connection to the host machine. WhereScape RED uses the Telnet connection in the drag and drop functionality. They are **not used in the actual production running of the Data Warehouse,** and is only necessary if you wish to use the drag and drop functionality.
Pre-Login Action	Response or command to send BEFORE logging in to the host machine. Typically this is NOT necessary but it can be used to indicate that the host Login Prompt is preceded by a line-feed (\n). However, it is preferable that the host login displays the Login Prompt without anything preceding it. [Optional]
Login Prompt	The host login prompt, or the tail end of the login prompt, e.g. **ogin as:**
Password Prompt	The host password prompt, or the tail end of the password prompt, e.g. **ssword:**
Post-Login Action	Not often used but may be necessary to respond to a login question. It is preferable that the host login goes straight to the command prompt.
Command Prompt	Enter the UNIX/Linux command prompt, or the tail end of that prompt, typically **>**. **Note** To ascertain some of the above fields, you have to log in to the UNIX system.
Big Data Adapter Settings	Set the two fields below to enable RED to communicate with BDA and enable loading data from Hadoop into Hive and/or into data warehouse tables using Sqoop. For further information about setting these fields, refer to **Connections to the Data Warehouse/Metadata Repository** and **Configuring the BDA Server**/**Configuring your database for use by BDA** for details.
Big Data Adapter Host	Host machine on which the Big Data Adapter is running its web-server.
Big Data Adapter Port	Port that Tomcat is running. Default is 8080.
Hadoop Connectors	Only available for Oracle databases using Oracle's Big Data Connectors. Refer to **Hadoop using Oracle's Big Data Connectors** for details.

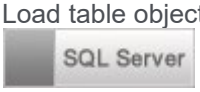
Credentials

Options	Description
UNIX/Linux User ID	User Account to login to the UNIX/Linux Host.
UNIX/Linux User Password	Password to login to the UNIX/Linux Host.
DSS User ID	Database user to connect to the WhereScape RED metadata repository.
DSS User Password	Database password to connect to the WhereScape RED metadata repository.

Other

Options	Description
Default Path for Browsing	Optional default Path for browser pane filter. When a path has been selected in this field, it becomes the initial point for browsing and it is also expanded on open in the right hand browser pane.

© Copyright, WhereScape Software Ltd All rights reserved.

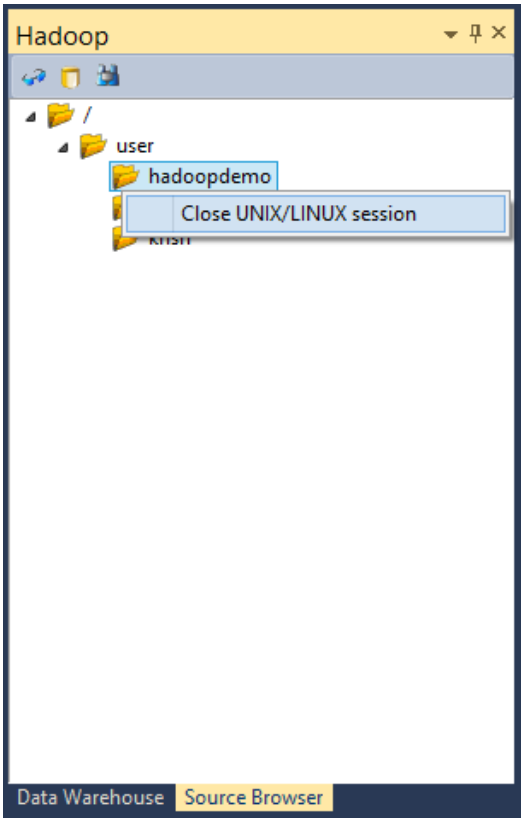
Options	Description
New Table Default Load Type	The default Load type for new tables created using this connection. Select from the **Script based load**, **Native SSH** or **Externally** loaded options. **Note** The available options in this drop-down list is configured from Tools > Options > **Available Load Types**.
New Table Default Load Script Template	The default **Script Template** to use when a **Script based load** type is defined for a Load table object that is **sourced** from this connection.  **SQL Server Hadoop Native SSH Loads** Please note that for SQL Server Hadoop Native SSH loads to be processed successfully, you must be running RED and the RED Scheduler on same machine as SQL Server to ensure that the files are accessible from the same path.
Data Type Mapping Set	XML files have been created to store mappings from one set of data types to another. Setting this field to **(Default)** will cause RED to automatically select the relevant mapping set; otherwise you can choose one of the standard mapping sets from the drop-down list or create a new one.

To test the drag and drop functionality

- From the menu strip select **Browse > Source Tables**
- Drill down to the area required
- Drag an item to the middle pane, (having first selected the object in the left pane).

Closing the Connection

To close the collection, right-click in the browser pane and select **Close UNIX/LINUX session**:



© Copyright, WhereScape Software Ltd All rights reserved.

Hadoop using Oracle's Big Data Connectors

WhereScape RED can load data from Hadoop into an Oracle repository using Oracle's Big Data Connectors **Oracle SQL Connector for Hadoop (OSCH)** and **Oracle Loader for Hadoop (OLH).**

RED loads data from Hadoop using Oracle's Big Data connectors via a Hadoop connection on UNIX/Linux from which users can do script based loads using the drag and drop functionality.

For more details about loading data from Hadoop after the connection is set up in RED, refer to **Flat File Load** and Flat File Load - Source Screen.

The following two sections describe the connection properties from within RED, using either the **Oracle SQL Connector for Hadoop (OSCH)** or the **Oracle Loader for Hadoop (OLH)** connectors.

To do loads from Oracle using the **OSCH** or **OLH** connectors, users will need to have the following system prerequisites before setting up a connection within RED:

- Oracle on Linux/Unix Hadoop tools installed on the Oracle Server and configured to connect to the Hadoop client
- **OLH** installed
- **OSCH** and/or **OLH** configured to connect to the Hadoop cluster and the Oracle database
- Hadoop client installed
- Oracle Wallet created and configured
- For **OSCH** - Oracle Directory Objects configured for the database user
- User permissions

Note

For scheduler configuration settings for the database user refer to Unix Scheduler for Native Loads using Oracle's Big Data Connectors in the RED Installation Guide.

Hadoop for Oracle Connectors

This topic describes in greater detail the connection properties as they apply to **Hadoop** connections using either **Oracle's SQL Connector for Hadoop (OSCH)** or **Oracle's Loader for Hadoop (OLH)**.

If the connection to Hadoop will not be using Oracle's Big Data connectors, refer to the previous section - Hadoop.
In RED, Hadoop as a source works with connections to Hadoop on UNIX/Linux from which users can do script based loads. The connection must be set via a Secure Shell (SSH) protocol.

To know more about system prerequisites required for setting up a connection within RED using either of these connectors refer to **Hadoop using Oracle's Big Data Connectors** for details.

Oracle Wallet for OSCH

An Oracle Wallet must be configured and used for the OSCH connector to work.

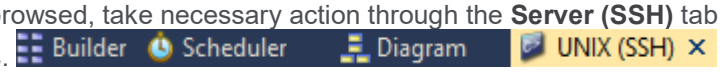
The Oracle user + SID used in the RED Connection object must be added to the Oracle Wallet.

If the Oracle user is not added to the wallet (or if it's added against a different SID, even if equivalent to the one used in the RED Connection object) then OSCH will fail with this message: "KUP04076: file name cannot contain a path specification."

Oracle Wallet for OLH:

RED only supports OLH with the Oracle Wallet, therefore an Oracle Wallet must be used with OLH.

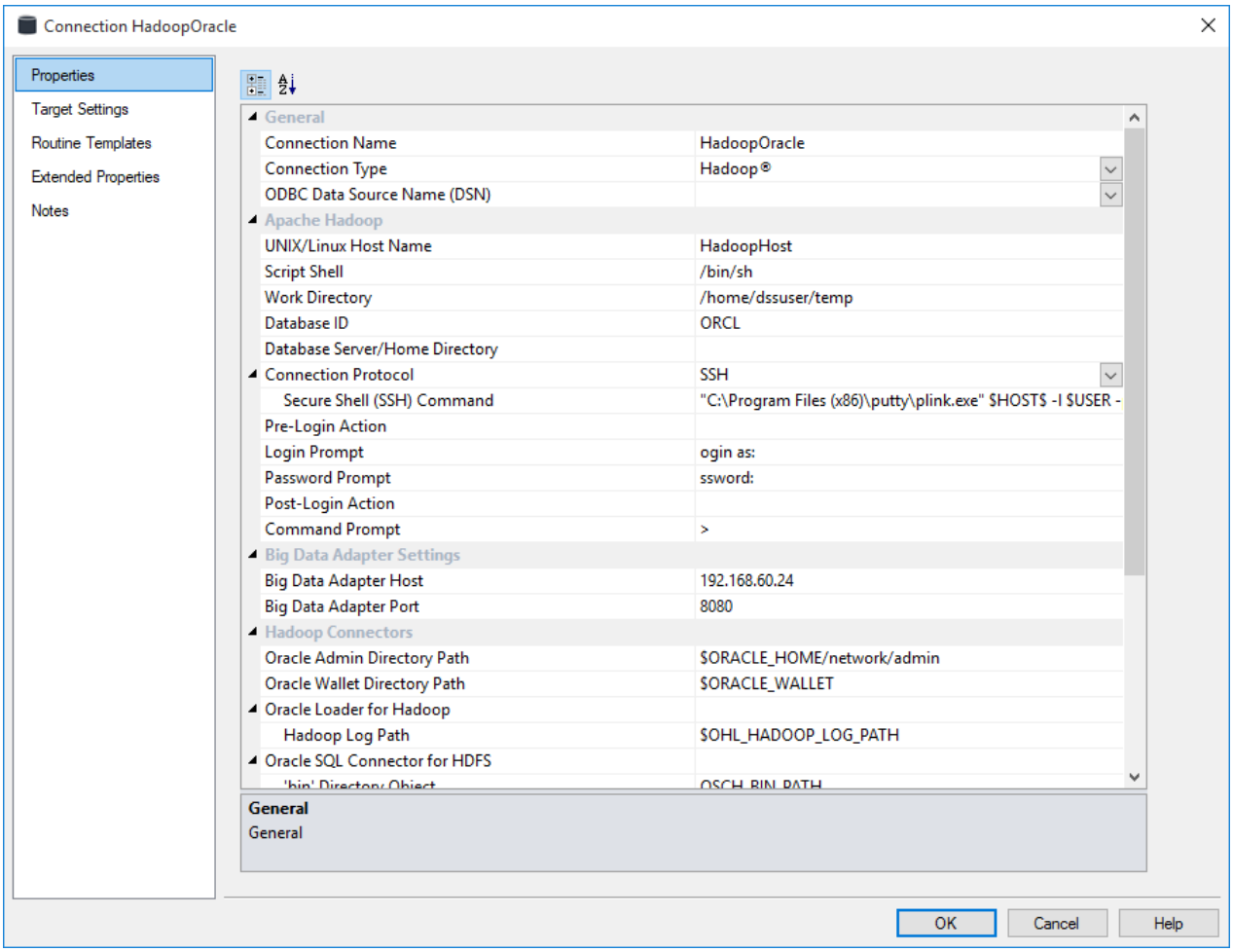
Although OLH does not require the use of the Oracle Wallet (unlike OSCH), the use of OLH without it is discouraged by Oracle. The Oracle user + SID used in the RED Connection object must be added to the Oracle Wallet.

If the **UNIX/Linux** connection returns a blank screen or an error message in the **Results** pane after the connection is browsed, take necessary action through the **Server (SSH)** tab next to the main Builder and Scheduler tabs. 

This tab is displayed after browsing the UNIX connection.

© Copyright, WhereScape Software Ltd All rights reserved.

Sample **Hadoop OSCH/OLH connection** screen:



General

Options	Description
Connection Name	Name used to label the connection within WhereScape RED.
Connection Type	Indicates the connection source type or the connection method, such as Database, ODBC, Windows, Unix. Select the **Hadoop** connection type.

Apache Hadoop

Options	Description
UNIX/Linux Host Name	IP address or host name that identifies the Hadoop server.
Script Shell	Path to the POSIX-compliant UNIX/Linux shell to use for generated scripts. For **UNIX** hosts, set to **/bin/ksh**. For **Linux** hosts set to **/bin/sh**. If this field is left blank, a default will be chosen based on the name of the connection and the type of database used for the WhereScape RED metadata repository.
Work Directory	Windows directory used by WhereScape RED to create temporary files for minimal logged extracts. The directory must exist and allow write access. There must be a different work directory for each WhereScape RED Scheduler running on the same machine to avoid file conflicts. Typically C:\Temp or a sub-directory of C:\Temp is used.
Database ID	Database Identifier (e.g. Oracle SID or TNS Name, Teradata TDPID) or Database Name (e.g. as in DB2 or SQL Server).

© Copyright, WhereScape Software Ltd All rights reserved.

Options	Description
Database Server/Home Directory	Optional to specify the Database Home Directory if it is different from the standard home directory.
Connection Protocol	Telnet or Secure Shell (SSH) protocol to use to connect to the UNIX/Linux machine. For SSH, the **Secure Shell (SSH) Command** property is enabled to specify how to connect.
Secure Shell (SSH) Command	Command to execute to connect to a Hadoop machine using the Secure Shell (SSH) protocol, such as **C:\putty\plink.exe -ssh some_host_name**.
Pre-Login Action, Login Prompt, Password Prompt, Post-Login Action, and Command Prompt.	These fields are only used to create a Telnet connection to the host machine. WhereScape RED uses the Telnet connection in the drag and drop functionality. They are **not used in the actual production running of the Data Warehouse,** and is only necessary if you wish to use the drag and drop functionality.
Pre-Login Action	Response or command to send BEFORE logging in to the UNIX/Linux machine. Typically this is NOT necessary but it can be used to indicate that the UNIX/Linux Login Prompt is preceded by a line-feed (\n). However it is preferable that the UNIX/Linux login displays the Login Prompt without anything preceding it. [Optional]
Login Prompt	The UNIX login prompt, or the tail end of the login prompt, e.g. **ogin as:**.
Password Prompt	The UNIX password prompt, or the tail end of the password prompt, e.g. **ssword:**.
Post-Login Action	Not often used but may be necessary to respond to a login question. It is preferable that the UNIX login goes straight to the command prompt.
Command Prompt	Enter the UNIX/Linux command prompt, or the tail end of that prompt, typically **>**. **Note** To ascertain some of the above fields, you have to log in to the UNIX system.
Big Data Adapter Settings	When using Oracle's Big Data Connectors to load data from Hadoop into Hive, it is not required to set the BDA fields..

Hadoop Connectors

Options	Description
Connector Type	Big Data Connectors that connect Hadoop with the Datawarehouse Database. **Oracle SQL Connector for HDFS** and **Oracle Loader for Hadoop** fields are available. Enter the relevant settings for each connector type in the fields displayed. **Note** To load data using one of these two specific connectors, the relevant connector must be selected from the **Hadoop Loader** field in the load table's **Source** screen. Refer to **Flat File Load - Source Screen** for more details.

Oracle Loader for Hadoop

Options	Description
Hadoop Log Path	Hadoop path to the log directory where a sub directory will be created with logs for each load run. This is the Hadoop path to main log directory. A sub directory is created for each job run and is then passed to OLH – if the directory passed to OLH does not exist or is not empty this will result in a job run failure. **Example** If "/user/oracle/hadoop/output/" is supplied in the connection object and the user then runs the job to load "hundredR.csv" file, the sub directory "/user/oracle/hadoop/output/WSL_hundredR_SEQ_RESTARTCOUNT/" will be created to store the logs, where "SEQ" is RED's sequence number and "RESTARTCOUNT" is the job's restart count.

© Copyright, WhereScape Software Ltd All rights reserved.

Oracle SQL Connector for HDFS

Options	Description
'bin' Directory Object	Oracle DIRECTORY object that points to the file location where the hdfs_stream exists. Oracle Directory object for the 'bin' directory of OSCH that must be installed on the Oracle server. This is best to be set up within the home directory of the user that runs Oracle but outside of ORACLE_HOME, e.g. if the Oracle user is 'oracle' then a good choice would be '/home/oracle/osch/<install_dir>/bin'). **Note:** The Hadoop client must be installed on the Oracle server. Oracle user must have read and execute permissions.
External Tables Directory Object	Oracle DIRECTORY object that points to the directory where OSCH location files live. Oracle Directory object for the directory where Oracle is to keep 'location files' (xml files that 'link' external table in Oracle and target file in Hadoop). This directory must be on the Oracle server, and it is best to set up within the home directory of the user that runs Oracle, but outside of ORACLE_HOME. e.g. If the Oracle user is 'oracle' then good choice would be '/home/oracle/osch/exttab'). The Oracle user should have read and write permissions.
Log Directory Object	Oracle DIRECTORY object that points to the OSCH log directory. It can point to the directory where the 'location files' are (it will point to the loader default directory if it is not specified). Oracle user should have read and write permissions.
Oracle Admin Directory Path	OS path to the directory that contains the 'tnsnames.ora' file. This is the full path to 'tnsnames.ora' – "$ORACLE_HOME/network/admin".
Oracle Wallet Directory Path	OS path to an Oracle wallet directory where the connection credential is stored. The full path to the Oracle Wallet location. It can be set to anything – e.g. "$ORACLE_HOME/network/admin" or "$ORACLE_HOME/wallets", but if the wallet is created on the Oracle server then it will likely be in Oracle ACFS when available (e.g. "/u02/app/oracle/wallet/"). **Example** To create an Oracle directory object named "osch_bin_path", pointing to "/home/oracle/osch/orahdfs-3.1.0/bin" directory and then grant read and execute permissions on that object to Oracle user "oschuser": CREATE DIRECTORY osch_bin_path AS '/home/oracle/osch/orahdfs-3.1.0/bin'; GRANT EXECUTE ON DIRECTORY osch_bin_path TO oschuser; GRANT READ ON DIRECTORY osch_bin_path TO oschuser; **Note:** It is possible to either provide values or provide names of UNIX environment variables in the connection object parameter values. If UNIX environment variables are used instead of actual values, note that UNIX/Linux scheduler must be restarted after any changes to these variables or those changes will not be available to any processes scheduled to run on that scheduler. Since these values should be very stable (they probably never change) this is only an issue during the environment setup—the administrator must remember to add those variables to the environment first before starting the scheduler. If the variables are ever changed then the scheduler needs to be restarted. All environment variables must start with $.

© Copyright, WhereScape Software Ltd All rights reserved.

Credentials

Options	Description
UNIX/Linux User ID	User Account to login to the UNIX/Linux Host.
UNIX/Linux User Password	Password to login to the UNIX/Linux Host.
DSS User ID	Database User to connect to the WhereScape RED metadata repository. Not required. All authentication is handled through Oracle Wallet since Oracle SQL Connector for Hadoop will not work without it.
DSS User Password	Database Password to connect to the WhereScape RED metadata repository. Not required. All authentication is handled through Oracle Wallet since Oracle SQL Connector for Hadoop will not work without it. **Note:** To perform loads from **Hadoop** using multiple schemas, the RED must be granted an extra set of privileges described in section **9.3 Creating an Oracle Dss User** of the RED Installation Guide.

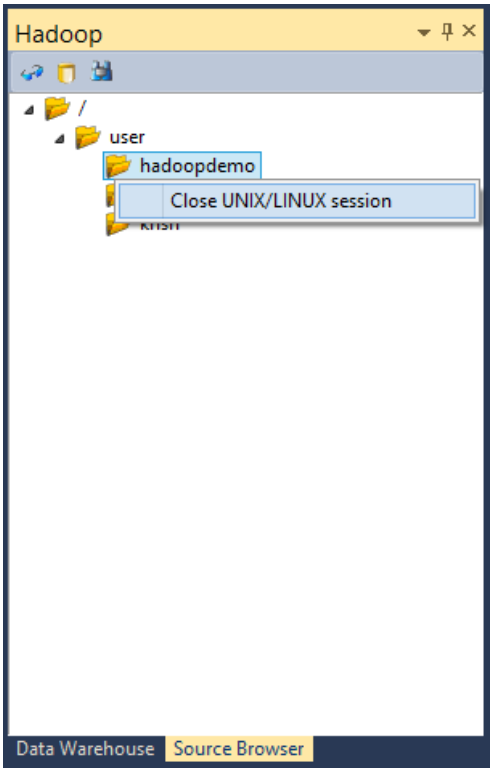
Other

Options	Description
Default Path for Browsing	Optional default Path for browser pane filter. When a path has been selected in this field, it becomes the initial point for browsing and it is also expanded on open in the right hand browser pane.
New Table Default Load Type	The default Load type for new tables created using this connection. Select from the Script based load, Native SSH or Externally loaded options.
New Table Default Load Script Template	The default **Script Template** to use when a **Script based load** type is defined for a Load table object that is **sourced** from this connection.
Data Type Mapping Set	XML files have been created to store mappings from one set of data types to another. Setting this field to **(Default)** will cause RED to automatically select the relevant mapping set; otherwise you can choose one of the standard mapping sets from the drop-down list or create a new one.
To test the drag and drop functionality	• From the menu strip select **Browse > Source Tables** • Drill down to the area required • Drag an item to the middle pane, (having first selected the object in the left pane).

© Copyright, WhereScape Software Ltd All rights reserved.

Closing the Connection

To close the collection, right-click in the browser pane and select **Close UNIX/LINUX session**:



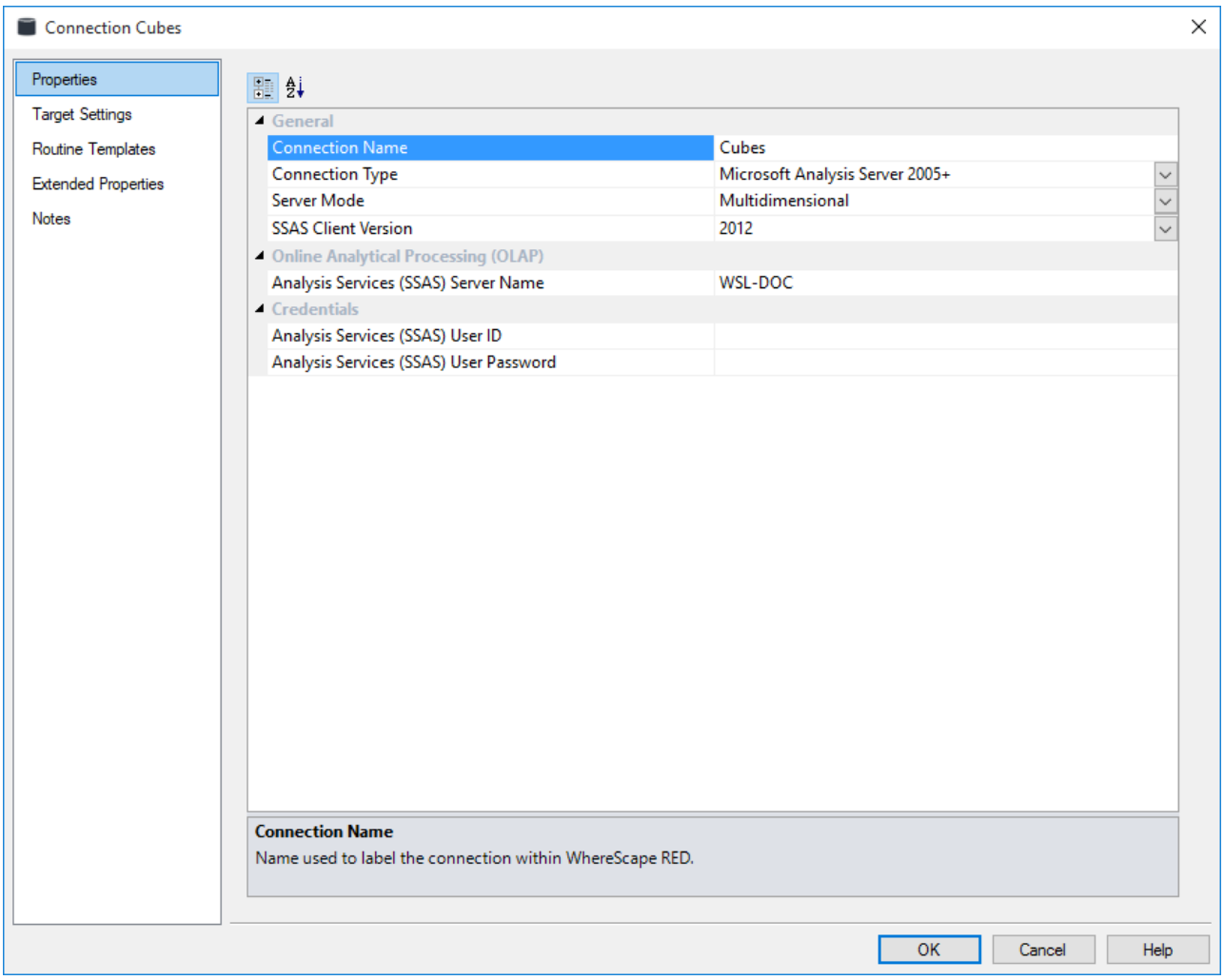
© Copyright, WhereScape Software Ltd All rights reserved.

Microsoft Analysis Server 2005+

Microsoft Analysis Server 2005+ - OLAP Cubes

A Connection to an **Analysis Services server** provides the location for cubes defined in the metadata repository. This connection is used in the creation and processing of cubes.

A sample screen is shown below:



General

Options	Description
Connection Name	Enter a name to identify the connection to the Analysis Services server.
Connection Type	Indicates the connection source type or method. Select Microsoft Analysis Server 2005+.
Server Mode	The operational mode that Microsoft Analysis Services will use. Select **Multidimensional** from the drop-down list.
SSAS Client Version	Microsoft Analysis Services Client version available for connecting to the SSAS database. It is recommended that the client version matches your database version. • If a 'Fail - missing AMO data provider' message is displayed in the Results pane when attempting to execute the OLAP action, check that the correct SSAS client version is specified and that the respective version of the data provider is installed on the client workstation.

© Copyright, WhereScape Software Ltd All rights reserved.

Options	Description
	Note If you have SSAS client version 2008 installed on your computer, WhereScape recommends selecting **2012** for the SSAS client version. If the required SQL Server Analysis Management Objects (AMO) are missing, refer to **https://msdn.microsoft.com/en-us/library/dn141152.aspx** for details.

Online Analytical Processing (OLAP)

Options	Description
Analysis Services Server Name	Enter the name of the Analysis Services server you wish to connect to.

Credentials

Options	Description
Analysis Services (SSAS) User ID and Password	User Name to connect to Analysis Services with when using SQL Server Authentication. Can be left blank for a trusted connection using Windows Authentication.
Analysis Services (SSAS) User ID and Password	Password to connect to Analysis Services with when using SQL Server Authentication. Can be left blank for a trusted connection using Windows Authentication.

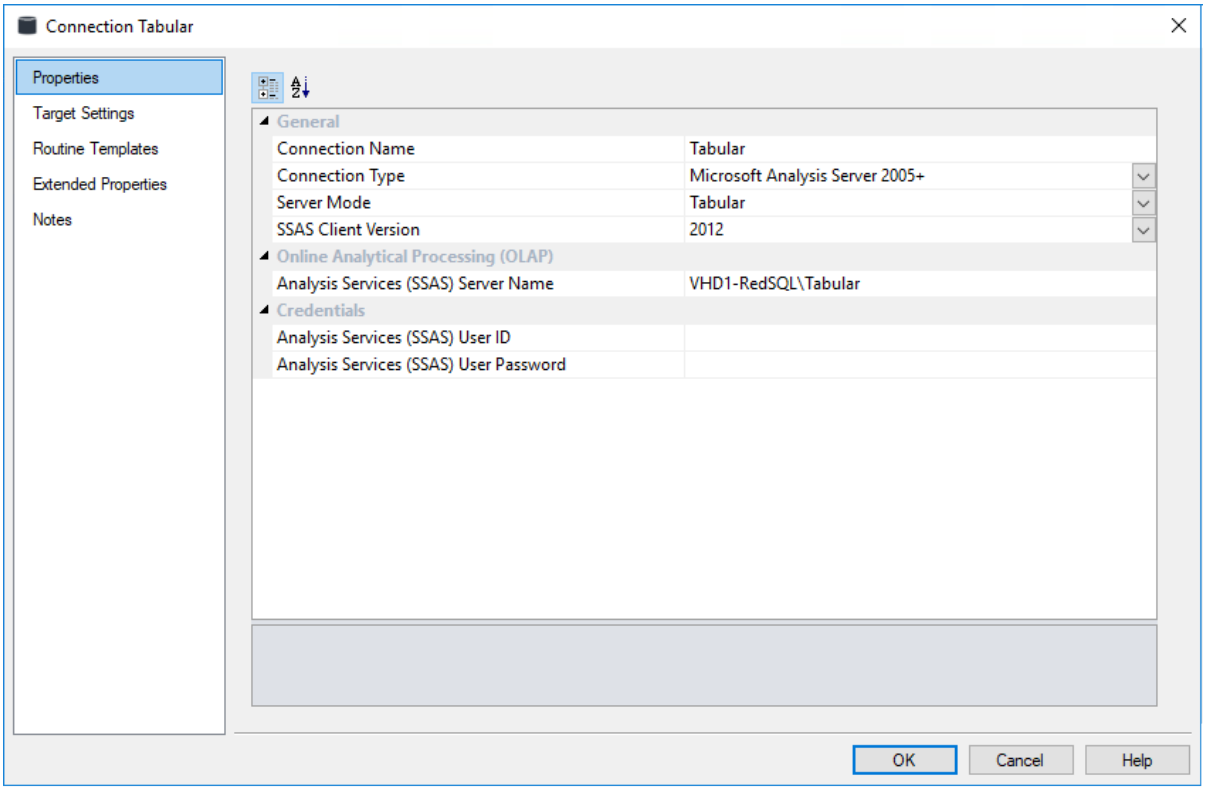
Microsoft Analysis Server 2005+ - Tabular Mode

A Connection to an **Analysis Services server** in **Tabular Mode** provides the location for Tabular cubes defined in the metadata repository. This connection is used in the creation and processing of Tabular cubes.

Note

Relationships must be created manually for tables stored on a Tabular target, for more information refer to **Relationship Maintenance** for details.

A sample screen is shown below:



© Copyright, WhereScape Software Ltd All rights reserved.

General

Options	Description
Connection Name	Enter a name to identify the connection to the Analysis Services server.
Connection Type	Indicates the connection source type or method. Select Microsoft Analysis Server 2005+.
Server Mode	The operational mode that Microsoft Analysis Services will use. Select **Tabular** from the drop-down list.
SSAS Client Version	Microsoft Analysis Services Client version available for connecting to the SSAS database. It is recommended that the client version matches your database version. If the required SQL Server Analysis Management Objects (AMO) are missing, refer to **https://msdn.microsoft.com/en-us/library/dn141152.aspx** for details.

Online Analytical Processing (OLAP)

Options	Description
Analysis Services Server Name	Enter the name of the Analysis Services server you wish to connect to. • You may need to specify the port number of the Analysis Services instance. To find your port number, follow the procedure documented in this Microsoft article: **https://support.microsoft.com/en-us/kb/2466860**. • An example of your Analysis Server (SSAS) Server Name using the port number in RED would be: `VH1D-REDSQL:49449\TABULAR`.

Credentials

Options	Description
Analysis Services (SSAS) User ID and Password	User Name to connect to Analysis Services with when using SQL Server Authentication. Can be left blank for a trusted connection using Windows Authentication.
Analysis Services (SSAS) User ID and Password	Password to connect to Analysis Services with when using SQL Server Authentication. Can be left blank for a trusted connection using Windows Authentication.

Target Table Location

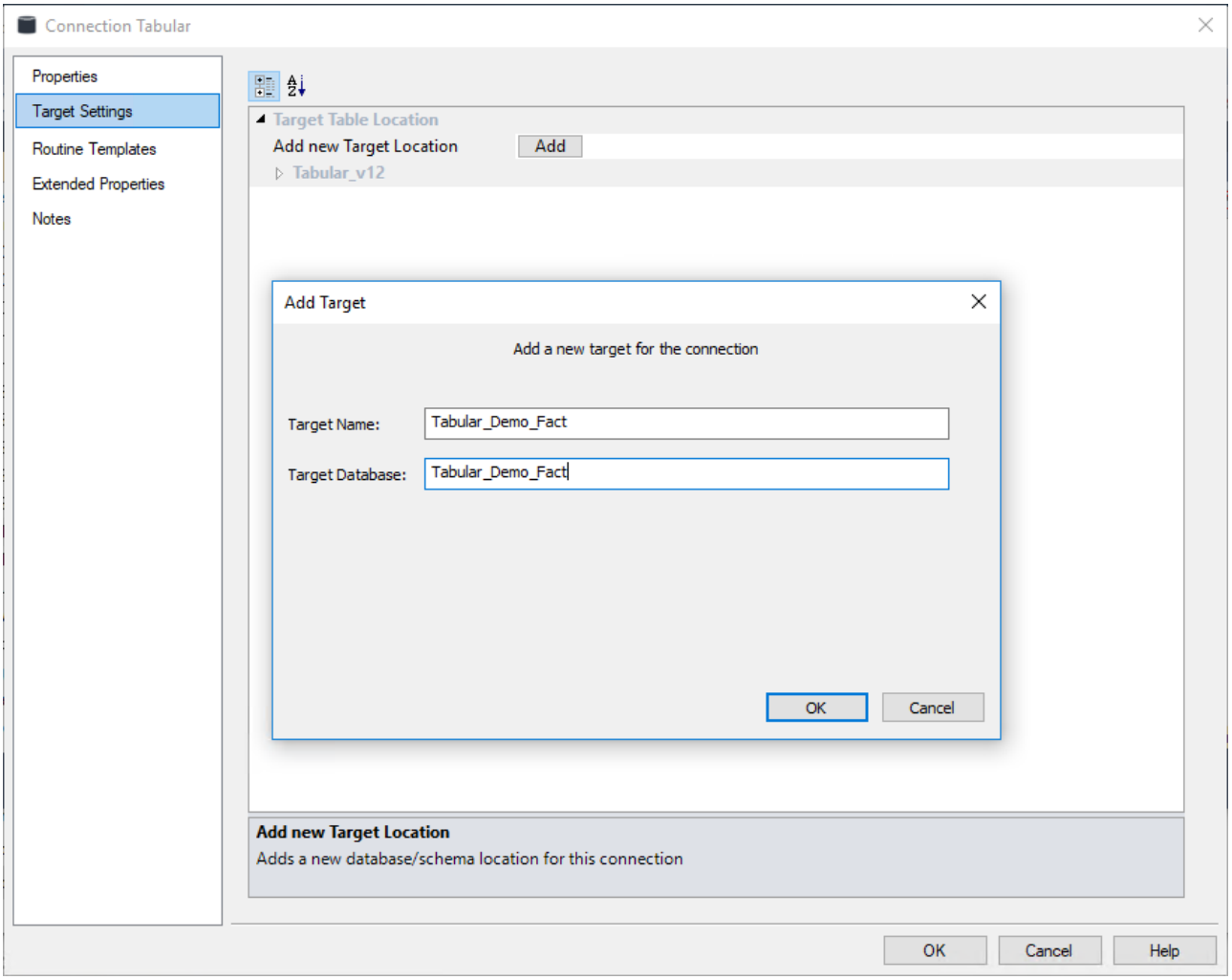
Note
To use the Tabular Mode functionality, targets must be created in the Tabular connection.

© Copyright, WhereScape Software Ltd All rights reserved.

Add new Target Location

Click the **Target Settings** tab to specify the name of the database associated with the targets to be used with this connection. To use the Tabular Mode functionality, it is required to create targets in the Tabular connection.

- The **Target Name** is the relevant Tabular Database's name displayed in RED.
- The **Target Database** is the relevant Tabular Database's name displayed in Analysis Services.



Browsing a Connection

The tables or files associated with a **Connection** can be displayed in the **Browser** pane on the right side of the **Builder** window by:

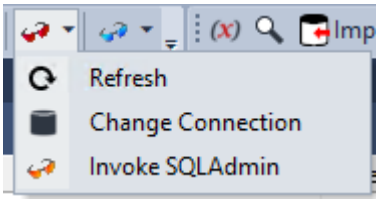
1. Selecting the **Browse > Data Warehouse** menu option to browse for the data warehouse connection,
2. Selecting the **Browse > Source Tables** menu option to browse a source system connection,
3. Right-clicking a Connection in the Objects list pane and selecting **Browse Connection** from the context menu, or
4. Clicking one of the two browser buttons from the toolbar:

The red button is used to browse the data warehouse connection and the blue button is used to browse a source system connection.

Each button remembers the last connection it browsed, so in this way one button can be used for the Data Warehouse and one for a source system.

© Copyright, WhereScape Software Ltd All rights reserved.

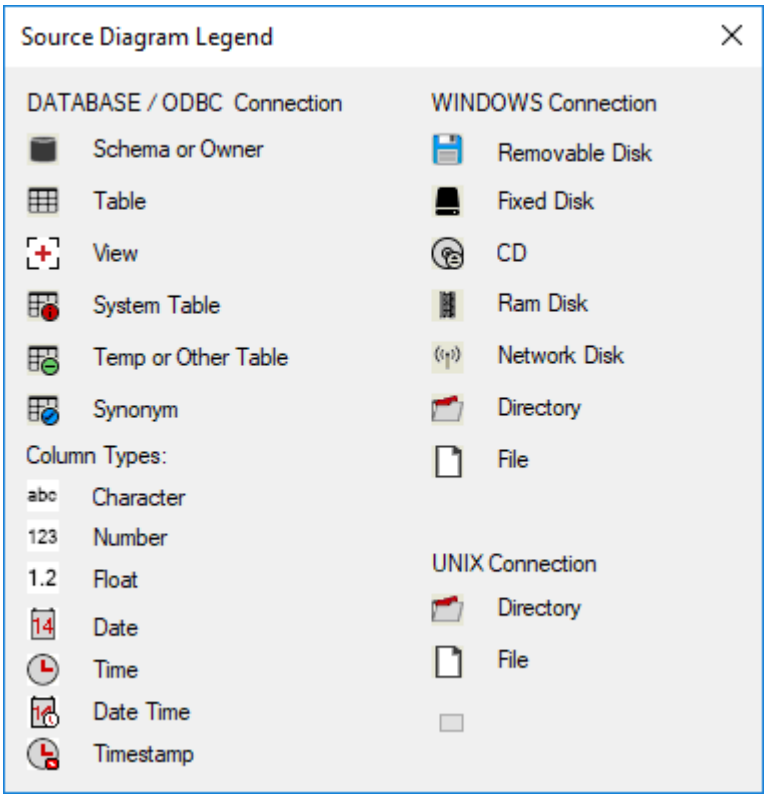
Clicking one of the buttons, displays the source tables without first displaying the source browser window. To change the connection, click the down arrow next to one of the browser buttons and then select **Change Connection**.



The current connection being browsed is shown in the status bar at the bottom right of the screen.

Browser Icons

When browsing a connection, the following legend applies for the source tables and objects.
The legend is displayed via the **Help > Source Legend** menu:



© Copyright, WhereScape Software Ltd All rights reserved.

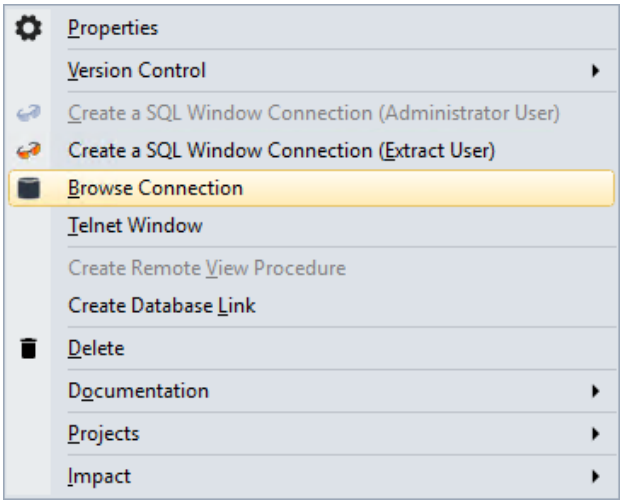
Connection Browse Properties

Tip

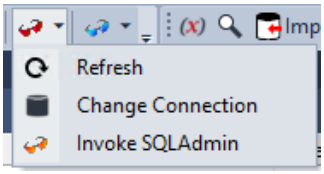
When browsing to a Connection, leave the schema field blank to see all schemas. To make RED browse a specific schema or schemas by default, go to a Connection's properties screen and enter the schema(s) to browse on the **Default Schema for Browsing** field.

To change the properties of the connection in the Browser pane:

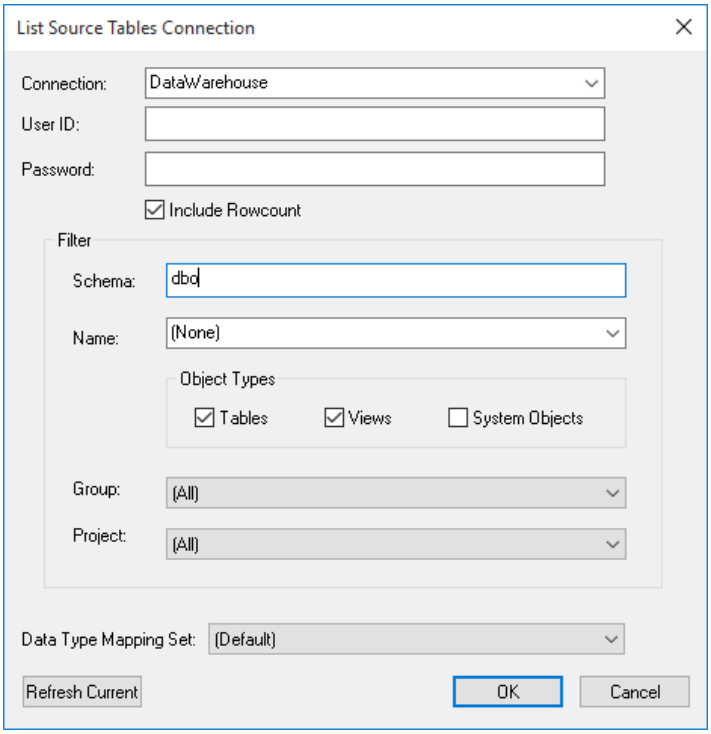
- Right-click a connection in the Objects list pane and select **Browse Connection**, or



- click the drop-down arrow next to one of the Browser buttons on the toolbar and select **Change Connection**.



The **List Source Tables Connection** window is displayed:



© Copyright, WhereScape Software Ltd All rights reserved.

This window enables you to change the properties of the connection you are browsing.

The **User ID** and **Password** fields can be changed in order to browse the connection as a different user.

A TDWallet string can be supplied as the password by selecting the **Password is TDWallet string** check box (this option only displayed in Teradata environments).

Selecting the **Include Rowcount** check-box displays a row count in brackets next to each source table in the Browser pane. This is only available for databases which update table statistics.

Note

For **Custom** database connections, the **Include Rowcount** option in the **List Source Tables Connection** screen is only available if the **Bulk Table Row Count SQL** field in the **Connection > Target Settings** tab has been defined. Refer to **Connection Target Settings for Custom Database** for details.

A **Filter** can also be applied when browsing a connection. Filters can be applied to any combination of:

- One or more Schema names (separated by commas),
- a standard SQL table name,
- specific Object Types (Tables, Views or System Objects),
- a Group, or
- a Project

The **Data Type Mapping Set** drop-down can be used to change the data type conversion used during drag and drop operations. If set to **(Default)**, the Data Type Mapping to use for each drag and drop operation is set based on the source and the Target Location selected. However, this can be changed in the **Add New Metadata Object** window, if required.

Changing a Connection's Properties

Whenever a connection's **Properties** are changed, the impact on the objects that use that connection must be considered.

Load tables have information from the connection stored within their **Properties**. This information is stored in the Load objects to minimize the complexity of the scheduled tasks.

The database link and database name are stored locally in each Load table. When either the database link or database name are changed on a connection, WhereScape RED displays the **Update Associated Load Tables** window.

Click **Yes** to automatically update the database link and/or database name on all associated Load tables.

This can also be done manually:

1. Double-click the **Load Tables** object group in the left pane. This displays all Load tables in the middle pane.
2. Select those Load tables that use the connection that has changed. Use the Windows standard keyboard keys (CTRL or SHIFT) for selecting multiple Load tables.
3. Right-click to open a context menu and then select **Change Connect/Schema**.
4. Select a different connection (e.g. Data warehouse) to change all the selected Load tables.
5. Repeat step (3) and now change the tables back to the altered connection. This updates all the Load tables with the new connection information.

Creating a Database Link

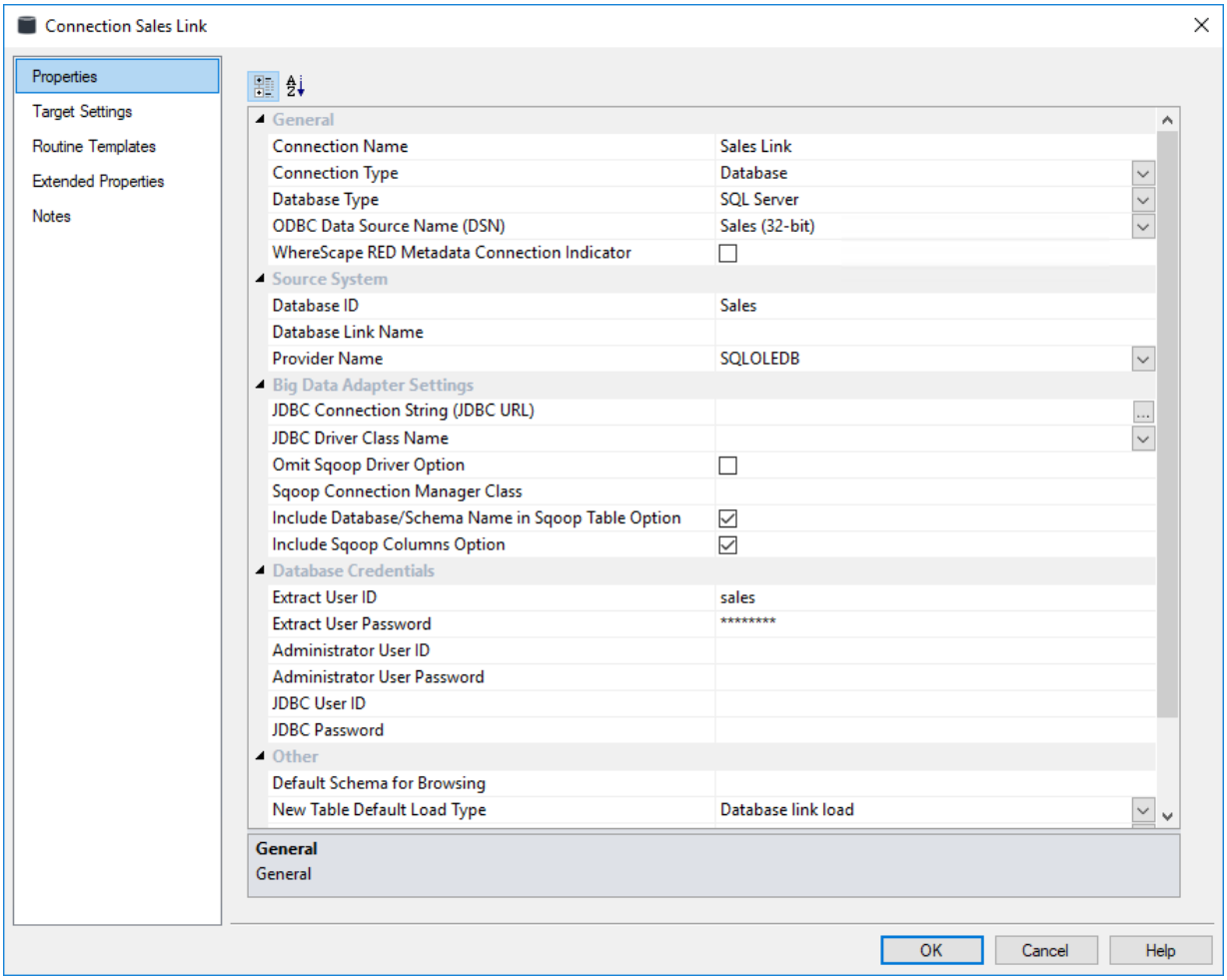
To create a database link connection, the database link must be established by clicking the **Create Database Link** option from the context menu, after the relevant connection has been created in RED.
Any existing connections can be used as a database link connection if:

- the connection type is **Database**;
- the **Database Link Name** field is populated with **a name** for the database link;
- the **Create Database Link** option has been selected from the connection's right click context menu. A confirmation message is displayed when the database link has been created.

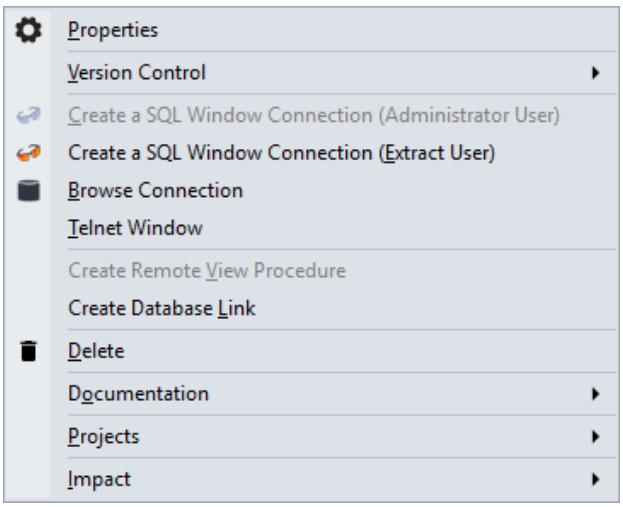
© Copyright, WhereScape Software Ltd All rights reserved.

To create a **new** Database Link type connection you need to do the following:

1. Enter a **name** for the new Connection.
2. Set the Connection Type to **Database.**
3. Select an **ODBC Source (DSN)** to extract the source data.
4. Populate the **Database ID** field with a valid Oracle SID or TNS Name for **Oracle** Databases.
5. Enter a **Database Link Name** for the database link. This is a required field for establishing the **Database Link** connection.
6. Enter the **Extract User ID** and **Extract User Password** for the user that has access to the source database when not using Windows Authentication.



7. Once a connection has been set up, you can right-click the connection name to see the following menu:

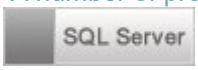


© Copyright, WhereScape Software Ltd All rights reserved.

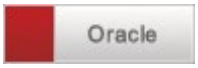
8. Once the **Create Database Link** option is selected, RED attempts to create a user database link to the source database. If the link already exists, a prompt appears to confirm if you want to overwrite the existing link.

Tips:

A number of problems can occur during this action, and your database administrator should be consulted:



For **SQL Server** a trusted link will be created if the extract user in the connection is set to **dbo**. In all other cases the extract user/password will be used as the log on for the remote server.



For **Oracle** data warehouses the more common problems are:

- Insufficient privileges to create the link. You need the **Create Database Link** privilege in order to create a new link.
- TNS could not resolve the connection to the remote database. The database link uses the value entered in the **SID** field of the connection to identify the remote database. This value must be a valid entry in the Tnsnames table for the source database.

Reset Meta Database Connections

From the **Help** menu, on RED's main top bar, users can select the **Reset Meta Database** Connections option.

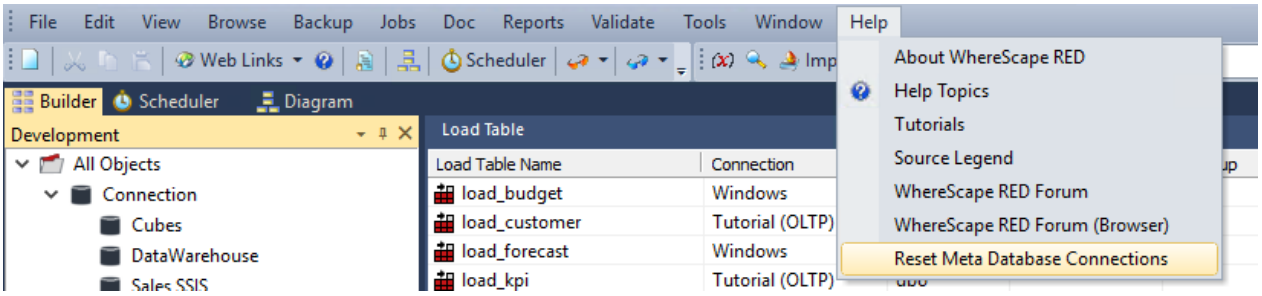
This option disconnects and frees most connections that RED has to any existing ODBC connections. This option can be useful for users that are already connected to existing ODBC sources but want to alter the credentials used.

When this option is used, RED attempts to release most existing ODBC connections so the next time an ODBC Connection is used, RED will attempt to reestablish a link.

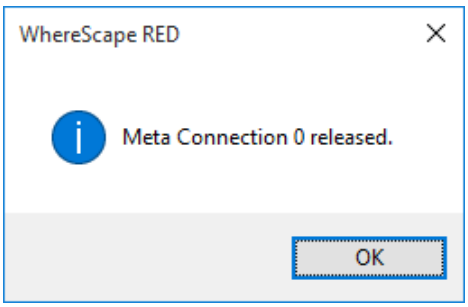
Note

At this stage, not all RED connections are handled via this mechanism and therefore not all connections will be reset when this option is used.

To reset meta database connections, click the **Help** menu in the main top bar and then click **Reset Meta Database Connections**.



Click **OK** on the following reset connection dialogs.



© Copyright, WhereScape Software Ltd All rights reserved.

Connection settings for BDA

This topic describes the required settings for connections using the WhereScape Big Data Adapter (BDA).

The WhereScape Big Data Adapter (BDA) is designed as an adapter to RED, focused on executing ELT related processing within the Hadoop/Hive eco-system.
For more information about the initial BDA setup, overview of BDA, the prerequisites and step-by-step instructions to set up BDA, refer to the **BDA** section of the WhereScape RED Installation Guide.

The BDA connection settings are always visible on both Hive and Hadoop connection types, which include the BDA server settings.

BDA settings are also displayed for any other database connection types if there is a Hive or Hadoop connection in the data warehouse, but these will only have the JDBC settings displayed.

Important

A Hive connection must exist before BDA settings (**Connection > Properties > Big Data Adapter Settings**) appear in other database connections.

BDA enables RED to use Sqoop as the load method to load data from Hive and HDFS into the data warehouse and also to load data into Hive as a target.

Only one BDA server connection is supported per Metadata repository.

Configuring the BDA Server

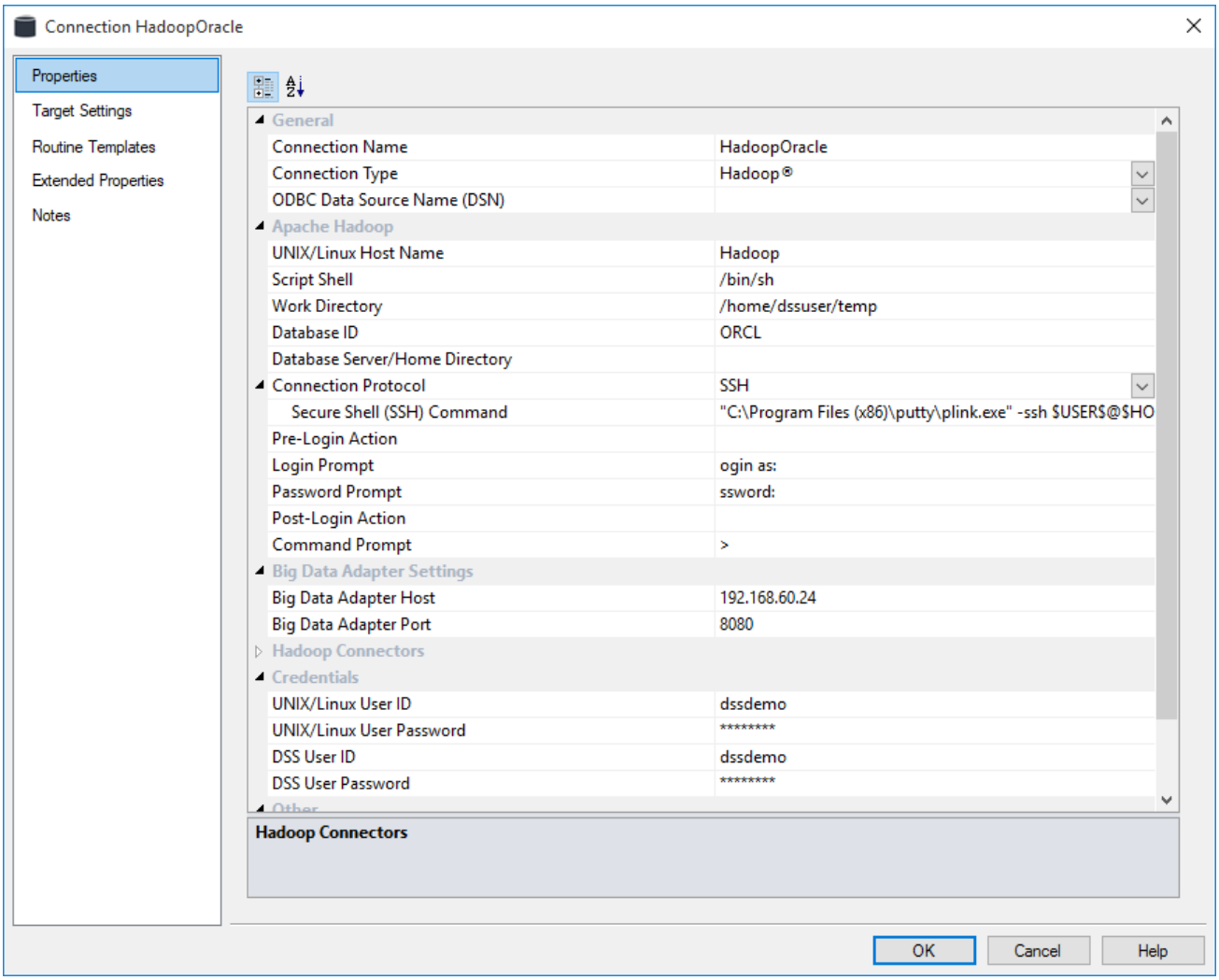
This topic explores the configuration of the BDA server in Hadoop connections to enable loading data from Hadoop into Hive and/or Datawarehouse tables as a source database into RED Datawarehouse tables using Sqoop.

For RED to be able to load data from Hadoop into Hive and/or Datawarehouse tables and also perform loads from Hadoop directly into the Datawarehouse using Sqoop, the **Big Data Adapter** settings need to be populated in Hadoop and Hive connections.

For more information about what is required for loading data into RED using Sqoop, see the **Big Data Adapter Settings** fields description below and also refer to **Connections to the Data Warehouse/Metadata Repository**, **Hadoop** and **Apache Sqoop Load** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

Hadoop connection example



Big Data Adapter Settings

Options	Description
Big Data Adapter Host	Host machine on which the Big Data Adapter is running its web-server.
Big Data Adapter Port	Port that Tomcat is running. Default is 8080.

© Copyright, WhereScape Software Ltd All rights reserved.

Configuring your database for use by BDA

Connections using BDA enable loading data into Hive as a target database and they also enable loading data from Hive into Datawarehouse tables as a source, using the Apache Sqoop load method.

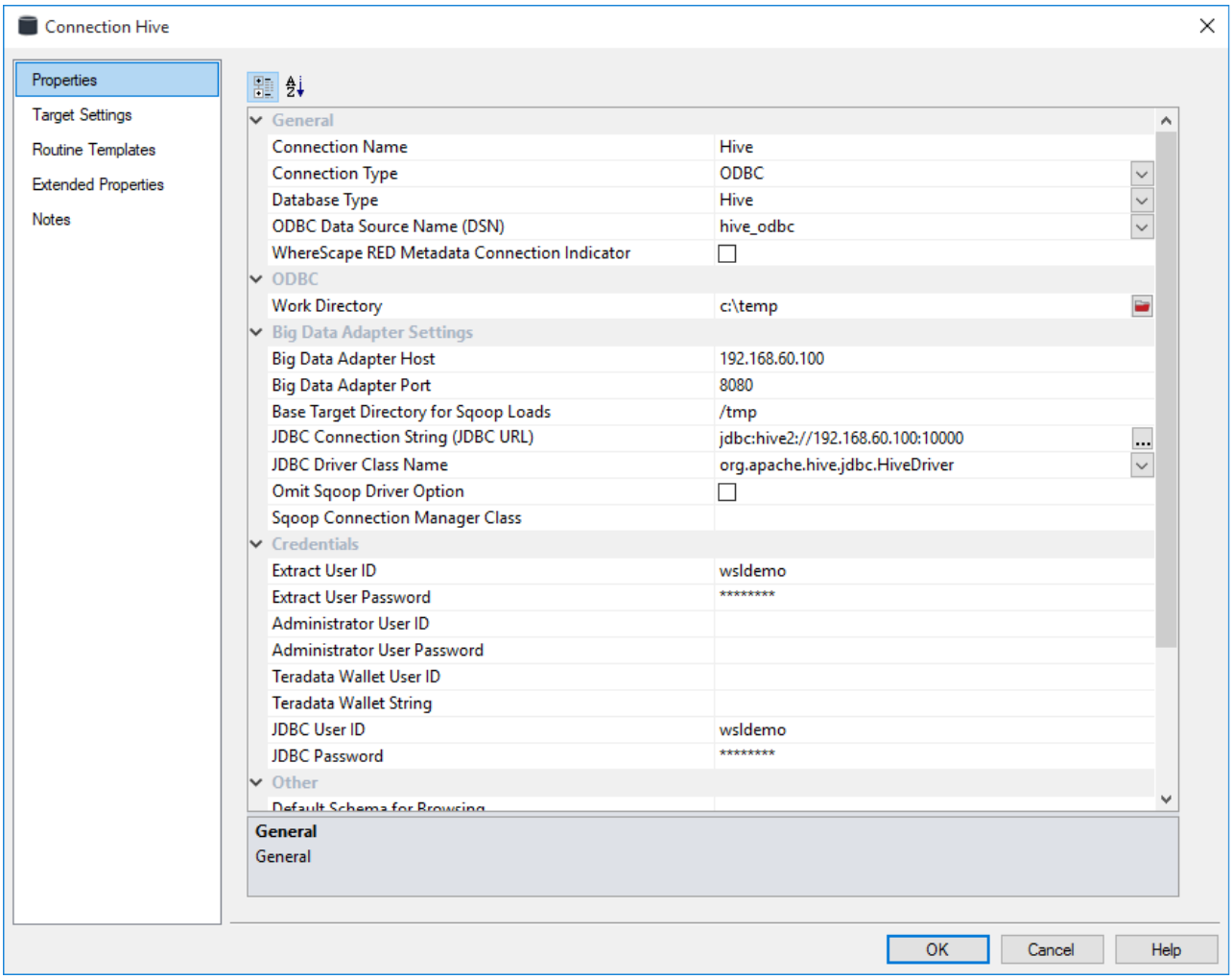
This connection type needs to include the JDBC connection string (JDBC URL) and related attributes (username, password) for the Hive database. The JDBC User and Password is usually the same as the Extract User ID but users can specify different credentials if necessary.

The Big Data Adapter settings will also need to be populated in the Datawarehouse connection. Refer to **Database - the Data Warehouse/Metadata Repository** for details.

RED can also load data directly into Hive from any database source. This load can also be processed via an Apache Sqoop load and the JDBC settings on the Hive connection will need to be populated. Please see the connection example and field description below for more details about this.
When loading data into Hive as a target, users can also add specific target locations in their Hive ODBC connections, if they have a Hive target license enabled.

Hive connection properties will be the same for any database sources.

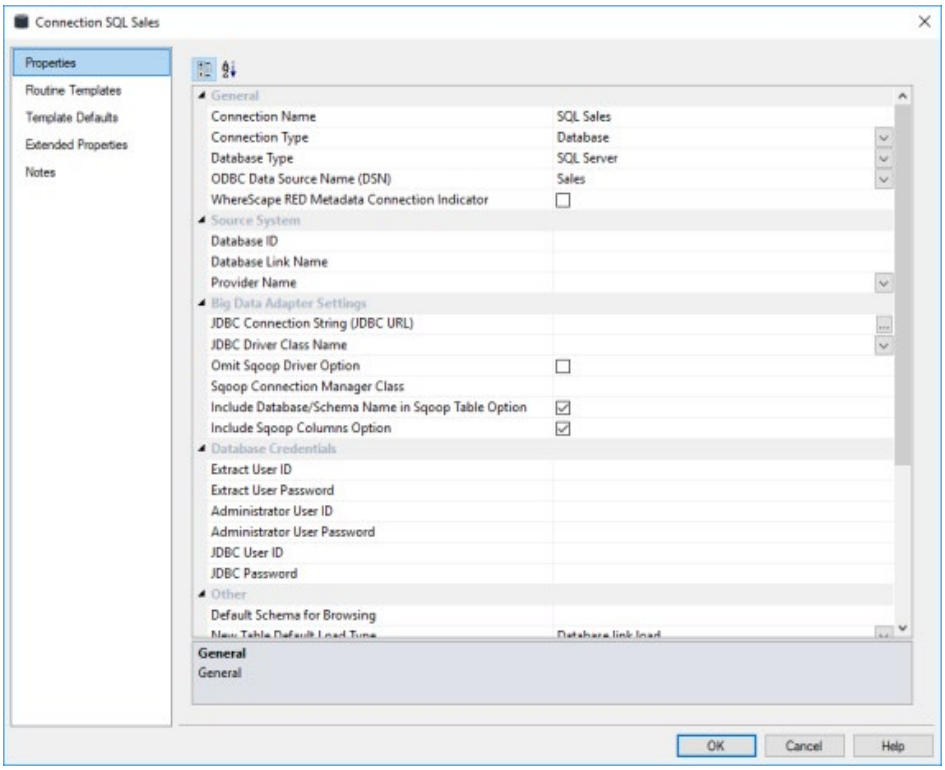
Example of a Hive ODBC connection



Big Data Adapter Settings

Options	Description
Big Data Adapter Host	Host machine on which the Big Data Adapter is running its web-server.
Big Data Adapter Port	Port that Tomcat is running. Default is 8080.
Base Target Directory for Sqoop Loads	HDFS directory in which to create target directories for Sqoop loads using the Big Data Adapter.

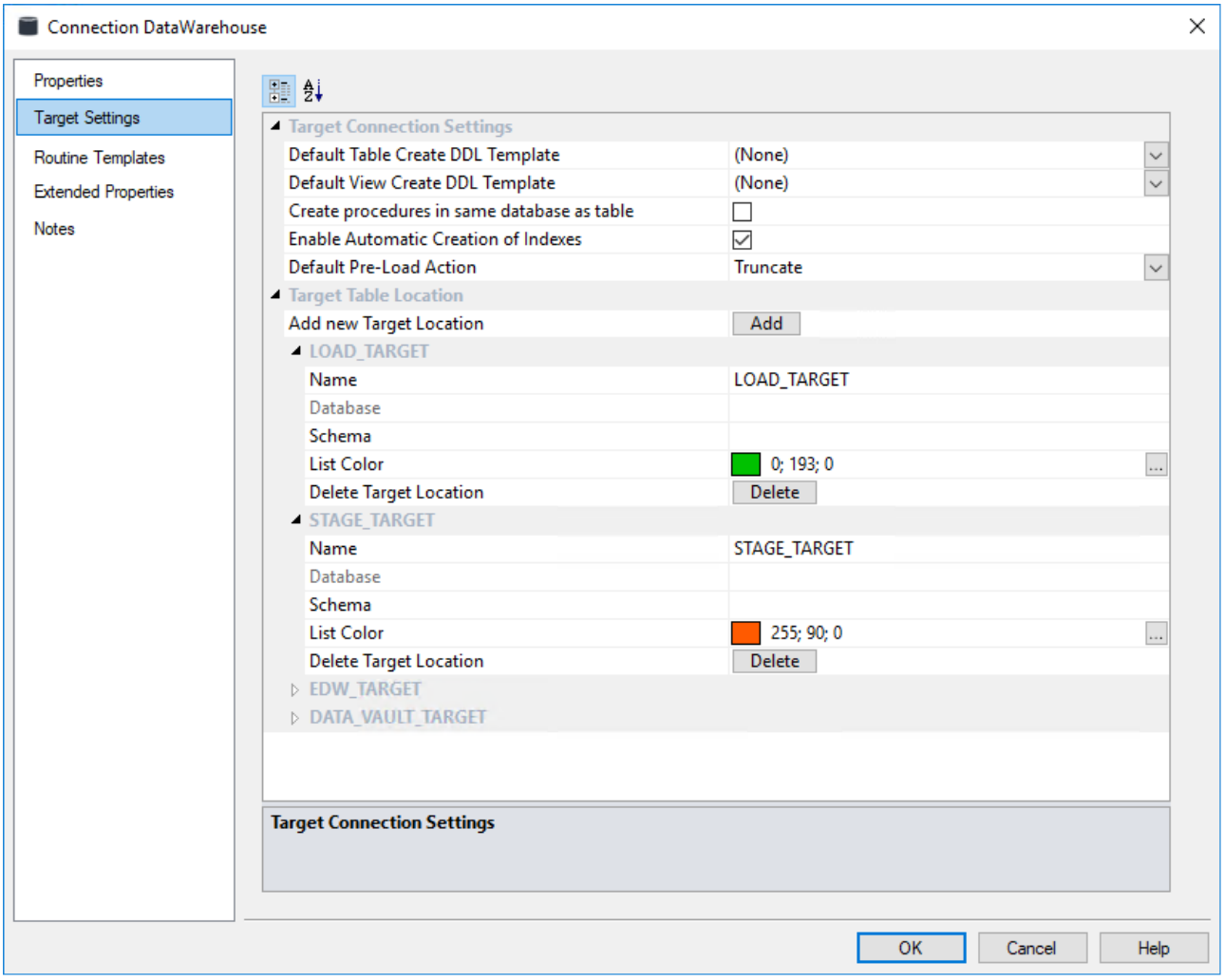
© Copyright, WhereScape Software Ltd All rights reserved.

Options	Description
JDBC Connection String (JDBC URL)	Connection string used by the WhereScape Big Data Adapter to access this database.
JDBC Driver Class Name	JDBC driver class to be used by the WhereScape Big Data Adapter. This field must be set if the JDBC URL is set. Select the appropriate JDBC Driver class name from the drop-down list. If this is left empty this will not be specified in generated commands.
Omit Sqoop Driver Option	If set, the --driver option to Sqoop is omitted. This is required for certain connection types such as Oracle connections. If you select the **Omit Sqoop Driver Option** check box, the driver parameter is not used in the Sqoop command line. This is a requirement for Oracle at the moment, as suggested by Sqoop documentation for 1.4.5.
Sqoop Connection Manager Class	Custom Sqoop connection manager class. Corresponds to the --connection-manager command line argument. Leave blank if this is not required. **Example connection from a source database to Hive** **The full JDBC connection string is: jdbc:sqlserver://192.168.60.100/DATABASE=SQL_Hive,DBS_PORT=1025** 

© Copyright, WhereScape Software Ltd All rights reserved.

Connection Target Settings

These settings enable you to define target connection settings and target table locations.



Target Connection Settings

Options	Description
Default Table Create DDL Template	The default Template used for generating DDL for new Tables created on targets of this connection.
Default View Create DDL Template	The default Template used for generating DDL for new Views created on targets of this connection.
Create procedures in same database as table	This setting enables you to specify if the procedures associated with new Tables created on Targets of this connection is created in the same database as the Table objects, or in the RED repository meta database. This option is not selected (OFF) by default for existing RED repositories, e.g. the Table's stored procedures are created in the RED repository meta database. If the Table is created in a different Target location (e.g. distinct schema/database in the same SQL Server instance), you can turn ON this option if you want the stored procedures to also be located in the same Target database as the Table object. Refer to **Distributing Tables across Multiple SQL Server Databases** for details. **Notes:** • This setting only applies for SQL server Target databases. For new RED

© Copyright, WhereScape Software Ltd All rights reserved.

Options	Description
	repositories, this option is ON by default (e.g. procedures are created in the same location as the Table object). New or regenerated procedures are created in the specified location. • Changing this setting has no impact on existing stored procedures. You must **Rebuild** or **Regenerate** the stored procedures for the changes to take effect. • If the **Lock Storage Location** option in the **Procedure Properties** window is set, then the current location of the procedure is retained, e.g. no changes will occur if the Target location of the procedure is manually changed and a **Rebuild** or **Regenerate** operation is performed via the Table object's **Properties** window or from the Table object's right click **Code** context menu, • If the Target location of a Table object is changed, then both the **Re-creation** of the Table object (as in previous versions of RED) and the **Regeneration** of the Procedure object is required for both objects to be moved. Refer to **Procedure Migration** for details.
Enable Automatic Creation of Indexes	This setting enables you to turn ON/OFF the automatic creation of metadata for indexes when creating new objects. Neither the RED metadata nor the physical index is created, if this setting is not enabled for objects. This option is selected (ON) by default. You can turn OFF this option for databases that do not have indexes, and if you do not require RED to automatically create metadata indexes for objects. **Notes:** • This setting only applies when creating metadata indexes for objects and does not affect existing metadata index records. • Turning this option OFF does not delete existing metadata records or the physical index.
Default Pre-Load Action	The default Pre-Load Action to use for Load tables created on targets of this connection. Options include: • Truncate • Execute Pre-load SQL • Both Truncate and Execute Pre-Load SQL • No action If **Execute Pre-Load SQL** is selected, then the SQL statement to execute before the load takes place must be entered—refer to Pre-Load Action in **Database Link Load Properties Screen** for details.

Target Table Location [For target enabled licenses]

Options	Description
Add new Target Location	This option enables you to add new database/schema locations for objects in this connection. For Greenplum, Netezza and PDW schema target locations setup, refer to **Database source system - Local/Linked Servers** or **ODBC** connection topics, depending on the connection type chosen for the database source. 1. Click the **Add** button to add the required target schemas for this connection. 2. Give the new target a name and then enter the target's schema. 3. Default schema location(s) for **New Tables** can also be set from the **Tools > Options** menu. Refer to **Settings > Storage > Target Location** for details. For more details on setting specific SQL Server and Oracle on a table by table level refer to **Table Storage Properties** for SQL Server and Oracle specific examples.

© Copyright, WhereScape Software Ltd All rights reserved.

Notes:

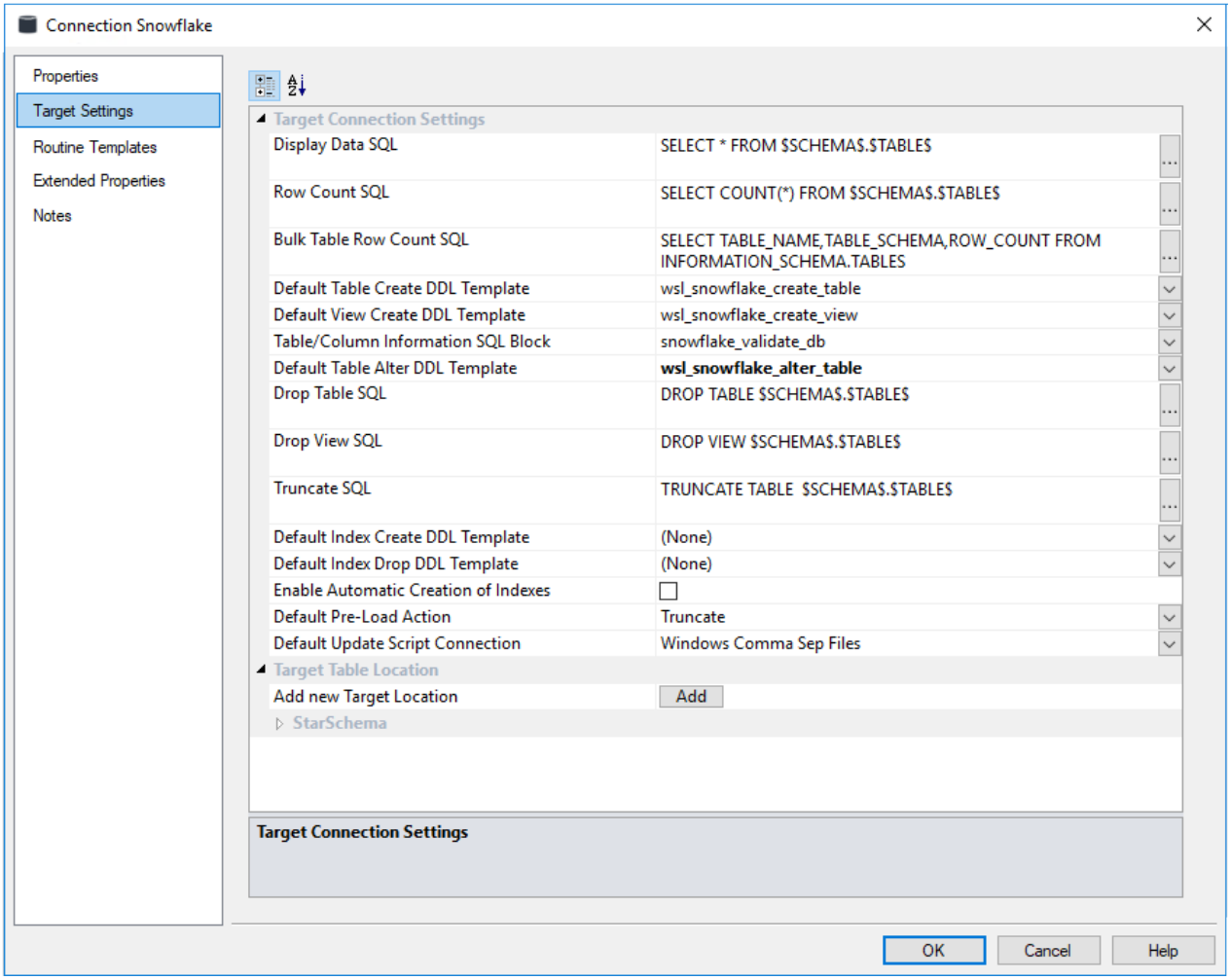
The database and schema names for **Custom** database connections are used as follows:
<database>.<schema>.object name
Leave database name blank if not required. Leave schema name blank to use the default schema.

Connection Target Settings for Custom Database

The **Target Settings** tab for custom database connections provide additional fields for defining default DDL templates to use for the creation of tables, indexes and views in custom database targets.

Note:

Please ensure that you have installed the WhereScape supplied templates or created your own templates, before configuring the settings below.

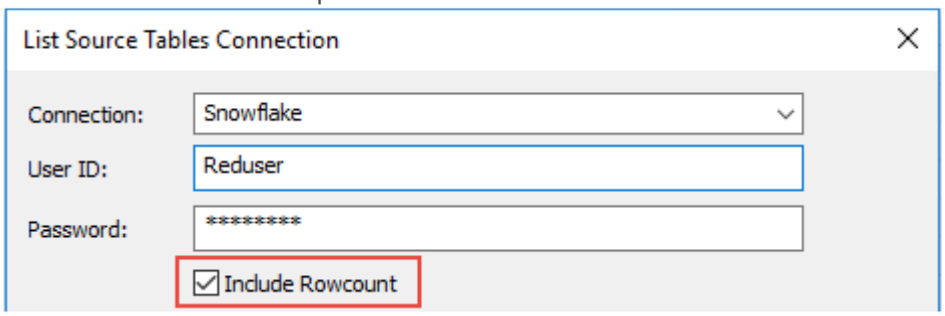


Target Connection Settings

In addition to the standard settings described in the previous section, the following settings are available for custom database connections:

Field	Description
Display Data SQL	The generic syntax of a Display Data SQL statement which should use RED placeholder parameters (e.g. $OBJECT$, $MAXDISPLAYDATA$).

© Copyright, WhereScape Software Ltd All rights reserved.

Field	Description
	Note: The $MAXDISPLAYDATA$ parameter is derived from the **User Preferences > Common > Look and Feel > General > Maximum rows returned for Display Data** setting.
Row Count SQL	The generic syntax of a Row Count SQL statement which should use RED placeholder parameters (e.g. $OBJECT$).
Bulk Table Row Count SQL	The SQL statement used to return the table name, schema name and row count of all tables for the connection. For example: **SELECT table_name, schema_name, row_count FROM information_schema.tables**. The variable $DATABASE$ can be used. The SQL statement defined in this field is used in the **Browser Connection screen** if the **Include Rowcount** option is enabled.  **Notes:** • The **Include Rowcount** option in the Browser Connection screen for Custom database connections is only available, if this field has been defined. • If the **Include Rowcount** option is enabled for a Custom database connection, then the SQL statement defined in this field is used to return the list of tables, schema and row counts as the source of the row counts; instead of the SQL statement defined internally in RED. • The SELECT statement must follow the order "table_name, schema_name, row_count" as shown in the example above.
Default Table Create DDL Template	The default Template used for generating DDL for new Tables created on targets of this connection. **Note:** This template is used in the validation done for custom database targets during Application Load Processing performed in the WhereScape Setup Administrator. Refer to Compile New and Changed Procedures in the RED Installation Guide for details.
Default View Create DDL Template	The default Template used for generating DDL for new Views created on targets of this connection
Table/Column Information SQL Block	The SQL Block Procedure used to query Table/Columns information; should use RED placeholder parameters (e.g. $DATABASE$, $SCHEMA$, $OBJECT$).
Default Table Alter DDL Template	The default Template used for generating DDL to alter existing Tables created on targets of this connection

© Copyright, WhereScape Software Ltd All rights reserved.

Field	Description
Drop Table SQL	The generic syntax of a Drop Table SQL statement which should use RED placeholder parameters (e.g. $OBJECT$).
Drop View SQL	The generic syntax of a Drop View SQL statement which should use RED placeholder parameters (e.g. $OBJECT$).
Truncate SQL	The generic syntax of a Truncate SQL statement which should use RED placeholder parameters (e.g. $OBJECT$).
Default Index Create DDL Template	The default Template used for generating DDL for new Indexes created on targets of this connection.
Default Index Drop DDL Template	The default Template used for generating DDL to drop existing Indexes created on targets of this connection.
Enable Automatic Creation of Indexes	RED's default behavior is to create indexes when creating new objects.
Default Pre-Load Action	The default Pre-Load Action to use for Load tables created on targets of this connection. Options include: • Truncate • Execute Pre-load SQL • Both Truncate and Execute Pre-Load SQL • No action If **Execute Pre-Load SQL** is selected, then the SQL statement to execute before the load takes place must be entered—refer to Pre-Load Action in **Database Link Load Properties Screen** for details.
Default Update Script Connection	The default Script Connection for new Update Scripts in tables created using this connection. This setting is only displayed in connections which have a target database type for which Update Script functionality is enabled, e.g. Custom database type.

Notes:

The following can be used as RED placeholder parameters:
$OBJECT$
$DATABASE$
$SCHEMA$
$TABLE$

The DDL settings defined in the Table Properties > **Storage** tab overrides the DDL settings defined in the **Target Settings** tab for custom database connections above.

Target Table Location [For target enabled licenses]

Options	Description
Add new Target Location	This option enables you to add new database/schema locations for objects in this connection. For Greenplum, Netezza and PDW schema target locations setup, refer to **Database source system - Local/Linked Servers** or **ODBC** connection topics, depending on the connection type chosen for the database source. 1. Click the **Add** button to add the required target schemas for this connection. 2. Give the new target a name and then enter the target's schema. 3. Default schema location(s) for **New Tables** can also be set from the **Tools > Options** menu. Refer to **Settings > Storage > Target Location** for details. For more details on setting specific SQL Server and Oracle on a table by table level refer to **Table Storage Properties** for SQL Server and Oracle specific examples.

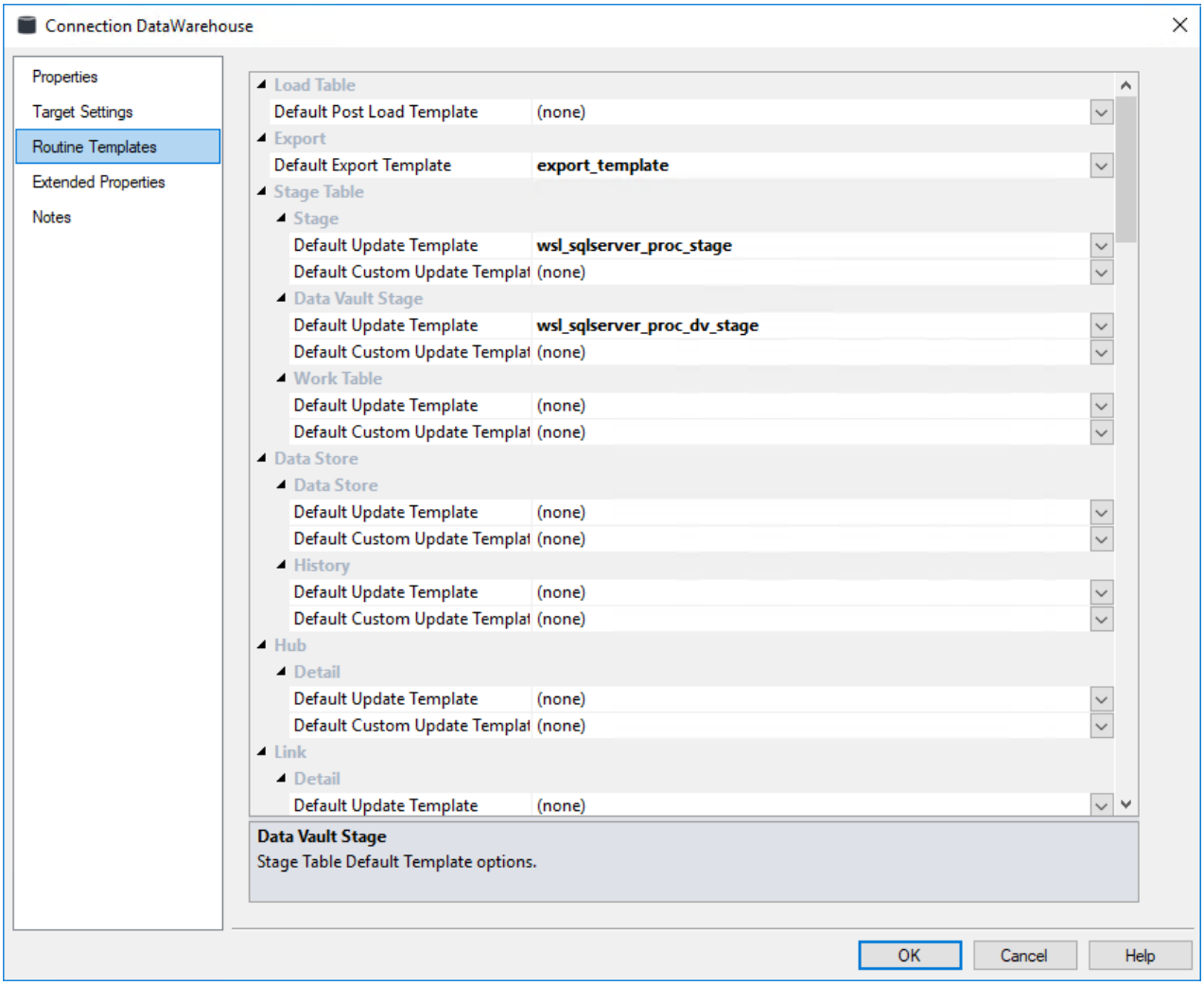
© Copyright, WhereScape Software Ltd All rights reserved.

Notes

The database and schema names for **Custom** database connections are used as follows:
<database>.<schema>.object name
Leave database name blank if not required. Leave schema name blank to use the default schema.

Connection Routine Templates

These settings enable you to set the default templates to use for the update routines of each table object associated with the selected connection. A default template can be set for the primary and custom update routines of each table by object type and sub type.



Notes:

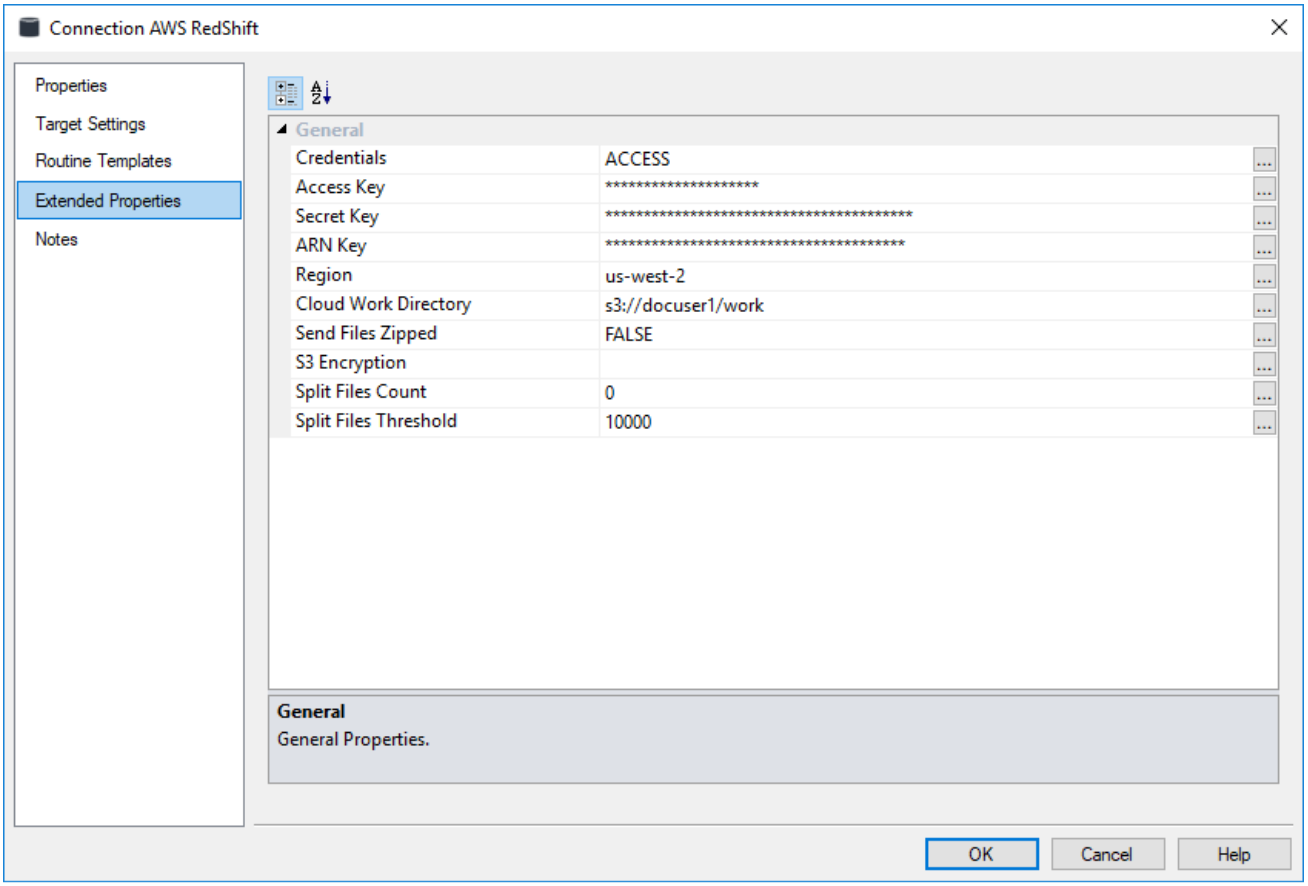
- Default Load Script Templates can be set in the source connection **Properties** tab.
- DDL Templates can be set in the **Target Settings** tab of the target connection.
- The settings in the Routine Templates tab is used in deploying applications into RED via the **REDCLI** command '`object generate-routine`'. Refer to **Dedicated Command Line Interface (REDCLI)** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

Connection Extended Properties

Note

These settings enable you to assign extended property values for a connection. Refer to **Extended Properties** for details.



© Copyright, WhereScape Software Ltd All rights reserved.

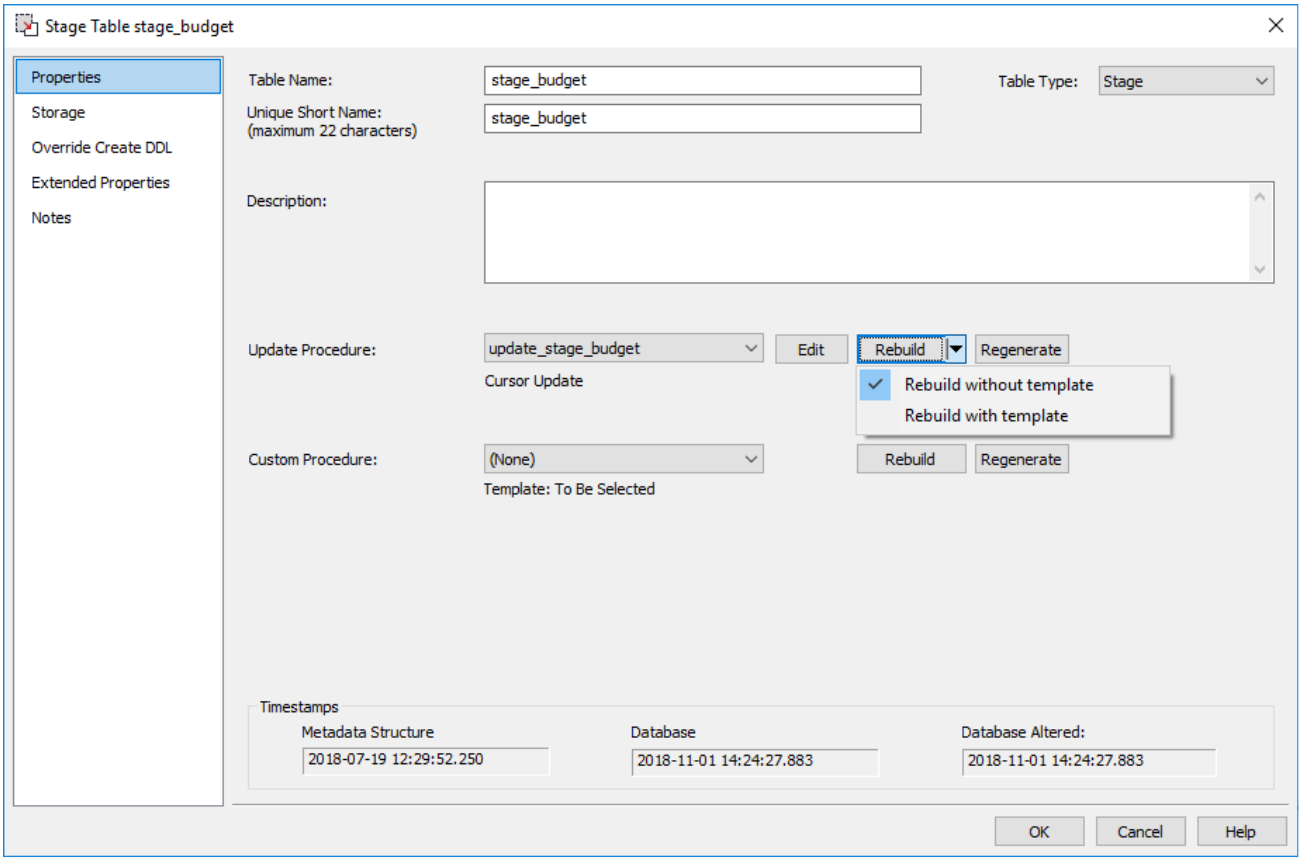
Table Properties

Various properties can be set on all table objects in WhereScape RED. The screens available in the Table Properties window depends on the object type selected and can be a subset of:

- Properties
- Storage
- Override Create DDL
- Source
- Documentation Fields
- Notes

Properties

The fields available in the **Properties** screen depends on the Object Type selected. More specific information is available in the respective sections that describe each RED object type.



Fields	Description
Table Name	The user-name of the selected table.
Unique Short Name	The short name is derived from the Table Name and is used internally by RED.
Table Type	The drop down provides a list of the available sub types for the selected table type.
Description	Optional, free text.
Update Type	**Note:** This field is only displayed if the table object is created in a custom database target, an additional processing option is available—script-based processing. Refer to **Script Templates for Custom Database Table Objects** for details. Enables you to select between **Procedure** or **Script** update type.

© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
	Selecting **Script** displays the **Update Script** drop-down field which enables you to select a script to use from the list of available script templates.
Update Procedure	The name of the procedure to be used when updating the table. **Note:** RED displays the name of the previously used update procedure template below the **Update Procedure** drop-down field by default, if the code generation is via a template.
Custom Type	Same as **Update Type** field above, this field is only displayed for table objects stored in a custom database target and provides option for **Procedure** or **Script** based custom processing.
Custom Procedure	The name of the procedure template to be used for custom update of the table. **Note:** RED displays the name of the previously used custom procedure template below the **Custom Procedure** drop-down field by default.
Edit	Edit the content of the displayed Update or Custom procedure.
Rebuild	Rebuild the defined Update or Custom Procedure. Refer to **Rebuilding Update Procedures** for details.
Regenerate	Regenerate the selected Update or Custom Procedure, using the responses to previously provided information.

Rebuilding tables using Update Procedures and Templates is applicable to the objects listed and described in the summary table, under the **Rebuilding Update Procedures > Generating an Update Procedure via a Template** section.

© Copyright, WhereScape Software Ltd All rights reserved.

Rebuilding Update Procedures

The update procedure for a data warehouse object can be generated by RED using parameters provided by the user, or it can be generated using a prepared template. Templates appropriate to the table type must first be created before they can be selected and used to generate update procedures. The option to select a template is only available if RED detects the presence of an appropriate template.

Notes:

- Update procedures for Table objects are stored in the RED repository meta database. However if required, the location of stored procedures can be changed and also be distributed across different Targets. The location of stored procedures can be changed and locked via the **Procedure Properties** window.
  Refer to **Procedure Migration** for details.
- Update procedures for Data Vault and Source Mapping objects can only be generated using a template.
  Refer to **Data Vault Templates** and **Generating Update Procedures for Source Mapping Objects** respectively for details.

Generating an Update Procedure without a Template

If RED does not detect a template, the drop-down options to rebuild with or without a template is not displayed in the **Rebuild** button. In this case, RED will manually regenerate the update procedure when **Rebuild** is clicked, by prompting the user to respond to a series of input requests. Refer to the **Generating Update Procedure** topic in each of the data warehouse object sections, for further information and the steps to manually generate update procedures.

Generating an Update Procedure via a Template

If RED detects the presence of a template, the **Rebuild** button provides additional options, in a drop down list, for rebuilding the selected update procedure. Two additional rebuild options are provided:

- **Rebuild without template** - RED rebuilds the update procedure, but ignores any previously used template. RED prompts you for inputs as required for the rebuild.
- **Rebuild with template** - Prompts you to select from a list, a template that is appropriate to the current object type. Depending on the selected template, the user is prompted to provide responses that will be used by the parameters within the template.

When you click the **Rebuild** button, RED rebuilds the procedure using the last selected option as the default method. If a template has not been previously used then RED prompts you for inputs as required for the rebuild.

Notes:

1. If a template has been previously used, RED uses this template by default when **Rebuild** is clicked. The name of this template is displayed below the **Update Procedure** field of the table **Properties** screen. The same applies to **Custom Procedures** that were generated via a template.
2. If you are using WhereScape RED on SQL Server or Oracle and you are licensed for the Star schema option, a template for Set based processing of Fact tables (e.g. wsl_sqlserver_proc_fact_set or wsl_oracle_proc_fact_set) is available from the RED **Template** objects list pane.
3. During procedure generation, a message prompt is displayed if any source tables are missing in the **Source Join** property of the **Source** tab in the **Update Build Options** screen. Select from the following options:
   - **Yes** to return to the **Source Join** screen and edit the **Source Join** property.
   - **No** to continue with the procedure generation.
   - **Cancel** to cancel the procedure generation.

© Copyright, WhereScape Software Ltd All rights reserved.

The following table summarizes the conditions that will display the drop-down options in the **Rebuild** button or that require a template. All other conditions will require a manual rebuild.

Table Object	SQL Server	Oracle	Teradata	DB2	Greenplum	Netezza	Hive/PDW/Custom
Stage	Both	Both	Both	Both	Both	Both	Template Only
Data Vault Stage	Template Only	Template Only	Template Only	Template Only	Template Only	Template Only	Template Only
Fact	Both	Both	Both	Both	Both	Both	Template Only
Fact Rollup	Both	Both	RED Automation Only	Both	RED Automation Only	RED Automation Only	Template Only
Fact KPI	Both	Both	RED Automation Only	Both	RED Automation Only	RED Automation Only	Template Only
Aggregate	Both	Both	Both	Both	RED Automation Only	RED Automation Only	Template Only
Dimension	Both	Both	Both	Both	Both	**Both**	Template Only
Data Store	Both	Both	Both	Both	Both	**Both**	Template Only
Normal, Hub, Link, Satellite, Custom1 or 2	Both	Both	Both	Both	Both	**Both**	Template Only

Both	This object type / database combination supports both RED automation for code generation and template based code generation.
RED Automation only	This object type / database combination uses RED automation for code generation and template based code generation is not currently available.
Template only	This object type / database combination only supports template based code generation.

Storage

The **Storage** screen of the **Table Properties** window displays the options applicable for storing data in the associated RDBMS.

The fields available on the **Storage** screen depends on the RDBMS on which you are storing the data:

For a SQL Server example, refer to **Table Storage Screen - SQL Server** for details.

For an Oracle example, refer to **Table Storage Screen - Oracle** for details.

For a DB2 example, refer to **Table Storage Screen - DB2** for details.

For a Greenplum example, refer to **Table Storage Screen - Greenplum** for details.

For a Netezza example, refer to **Table Storage Screen - Netezza** for details.

For a PDW example, refer to **Table Storage Screen - PDW** for details.

For a Teradata example, refer to **Table Storage Screen - Teradata** for details.

For a Custom Database example, refer to **Table Storage Screen - Custom Database** for details.

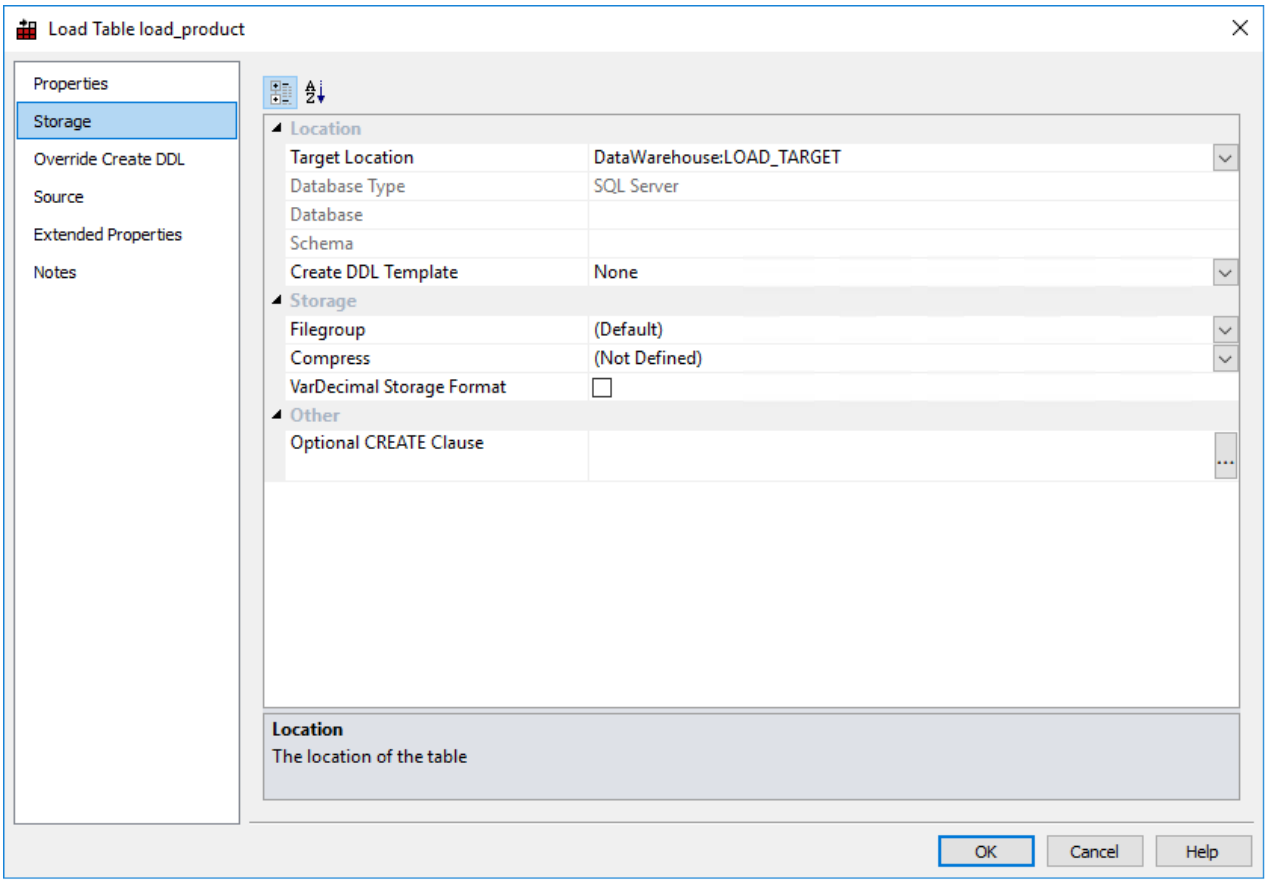
For a Tabular example, refer to **Table Storage Screen - Tabular** for details.

For information on changing storage locations for multiple tables at once, refer to **Bulk Table Storage Change** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

Table Storage Screen - SQL Server

Typical Storage screen for a **SQL Server** Table:



Location

Fields	Description
Target Location	The target location that defines the path of the table. Select (local) for a local table or select the target schema if you are locating tables in different schemas. **Note** The Force Target Usage setting in the **Tools > Options > Storage > Target Usage** removes the local target option from the drop-down list. Refer to **Target Usage** for details. • To add another database/schema to the list, refer to **Connections to the Data Warehouse/Metadata Repository** for details. • To set default target locations for tables, refer to **Settings > Storage > Target Location** for details. **Note** When upgrading from a RED version previous to 6.8.2.0 and moving existing objects to a target location, all procedures that reference those objects need to be rebuilt. Any FROM clauses also need to be manually regenerated in order for the table references to be updated to the new [TABLEOWNER] form.
Database Type	The database type for a connection that is used for target Data Warehouse tables.

© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
Database	**Note** This field is only visible if using multiple database as target locations for tables in an SQL environment. Refer to **Distributing Tables across Multiple SQL Server Databases** for details.
Schema	The schema where the table is located.
Create DDL Template	Optional. Specify the template to use when creating a new DDL procedure script. This option is only visible if a DDL template is available for this database type. Default value is *None.*

Storage

Fields	Description
Filegroup	Select a name from the drop-down list box. If no filegroup name is selected for SQL Server, the default for the database is used.
Compress	This field enables you to compress a particular table/index. Compression options are: • NONE • PAGE • ROW
VarDecimal Storage Format	Enables variable numeric storage in SQL Server for the table. The default is off - no variable numeric storage. To enable variable number storage, the following command must also be run for the SQL Server database: *sp_db_vardecimal_storage_format 'DatabaseName', 'ON'*

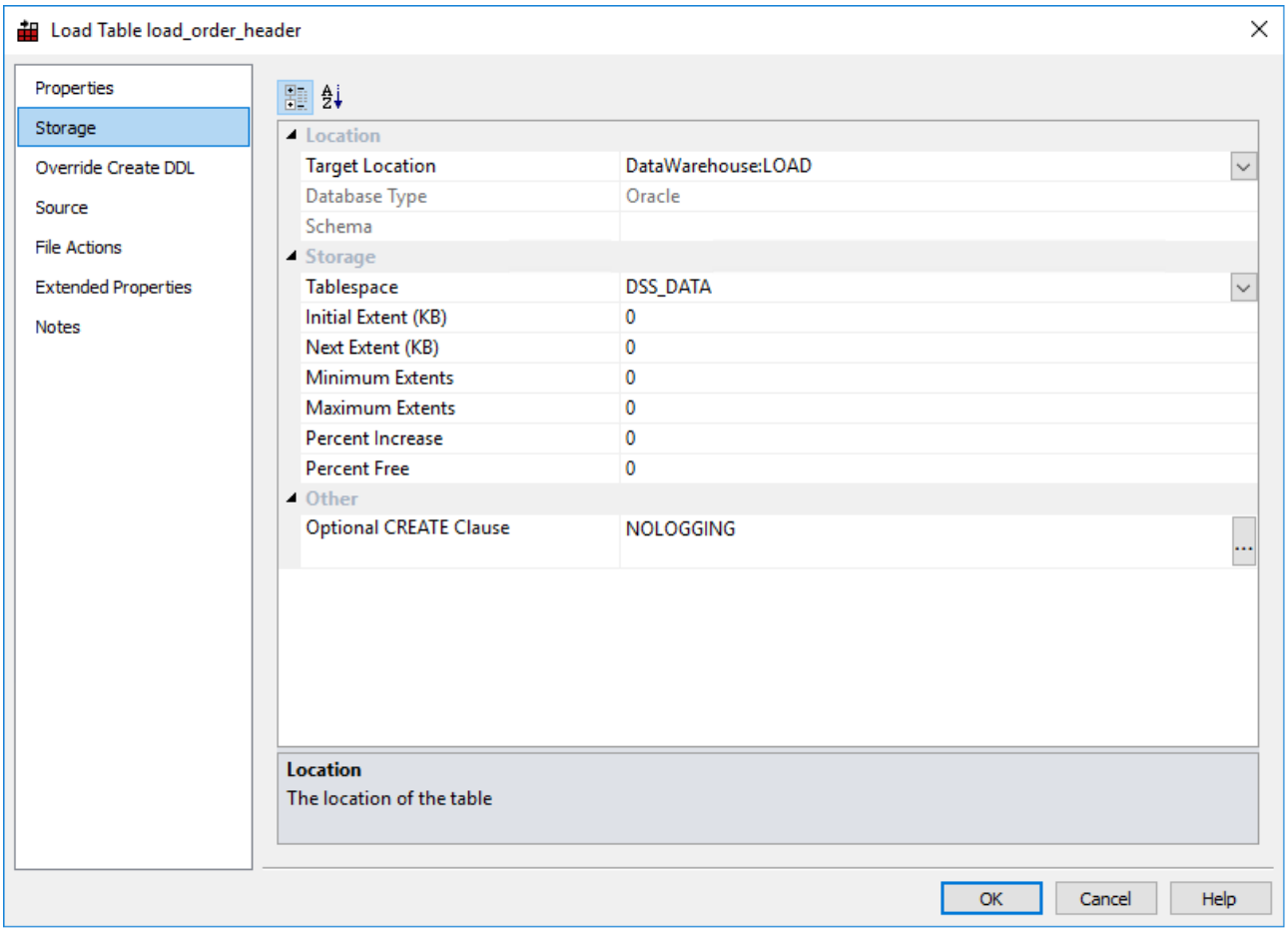
Other

Fields	Description
Optional CREATE Clause	Database-specific-and-compliant DDL to append to the generated CREATE TABLE statement. **Tip** To set default create clause values for all new objects, go to the **Tools > Options > Storage** menu - **Default Optional CREATE Clause**.

© Copyright, WhereScape Software Ltd All rights reserved.

Table Storage Screen - Oracle

Typical Storage screen for an **Oracle** table:



Location

Fields	Description
Target Location	The target location object that defines the path of the table. Select (local) for a local table or select the target schema if you are locating tables in different schemas. **Note** The Force Target Usage setting in the **Tools > Options > Storage > Target Usage** removes the local target option from the drop-down list. Refer to **Target Usage** for details. • To add another database/schema to the list see more details on Connections to the Data Warehouse/Metadata Repository (refer to **Database - Data Warehouse/Metadata Repository**). • To set default target locations for tables , refer to **Settings > Storage > Target Location** for details.
Database Type	The database type for a connection that is used for target DataWarehouse tables.
Schema	The schema where the table is located, if applicable.
Create DDL Template	Optional. Specify the template to use when creating a new DDL procedure script. This option is only visible if a DDL template is available for this database type. Default value is **None.**

© Copyright, WhereScape Software Ltd All rights reserved.

Storage

Note

When upgrading from a RED version previous to 6.8.2.0 and moving existing objects to a target location, all procedures that reference those objects need to be rebuilt. Any **FROM** clauses also need to be manually regenerated in order for the table references to be updated to the new [TABLEOWNER] form.

Warning

If the database type does not support moving tables, such as Oracle, Greenplum, Netezza, Teradata and PDW, all affected tables also need to be manually recreated after any storage changes.
Please note that changing the Storage for Dimension and Fact tables need to be handled very carefully, as artificial key relationships between Dimension and Fact could become out of sync.
Recreating Fact tables and large Dimension tables might take a considerable amount of time .

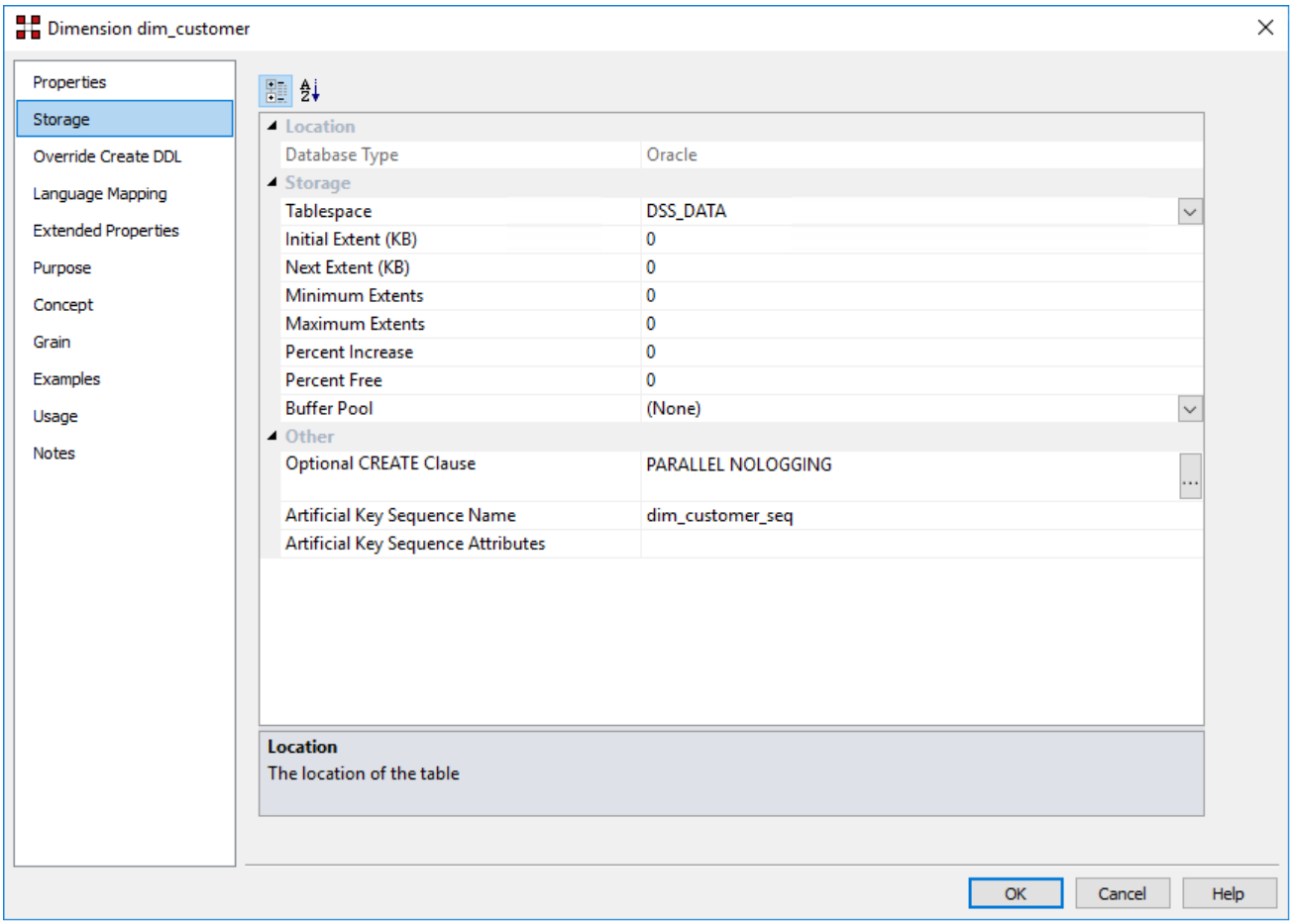
Fields	Description
Tablespace	Select a name from the drop-down list box. If no tablespace is selected in Oracle, then the default tablespace for the schema is used.
Initial extent (KB)	The size of the first extent of a created database table. The default is **0** kilobytes - use the tablespace's default value. For locally managed tablespaces, this field is ignored.
Next extent (KB)	The size of the next (second) extent of a created database table, unless percent increase is specified. The default is **0** kilobytes - use the tablespace's default value. For locally managed tablespaces, this field is ignored.
Minimum extents	The number of extents the table is created with. The default is **0** - use the tablespace's default value. For locally managed tablespaces, this field is ignored.
Maximum extents	The maximum number of extents the table can have. The default is **0** - use the tablespace's default value. For locally managed tablespaces, this field is ignored.
Percent increase	The size the next extent is as a percent of the previous extent added. For example, a value of 5 will create the next extent 5% bigger than the previous extent added. The default is **0** - use the tablespace's default value. For locally managed tablespaces, this field is ignored.
Percent free	The amount of free space to leave in each extent in Oracle. The default is 0 - use the database default.

Other

Fields	Description
Optional CREATE Clause	Database-specific-and-compliant DDL to append to the generated CREATE TABLE statement, such as a logging clause and/or a partition clause. • The default values for Oracle are NOLOGGING for Load Tables and PARALLEL NOLOGGING for remaining objects. • These values can be edited on the free text field of the **Storage** tab from every table's **Properties** screen. **Tip** To edit Oracle's default values for new objects, go to the **Tools > Options** menu - **Default Optional CREATE Clause**.
Buffer Pool	**Oracle Dimension tables only**: Mark the table to be retained in a special buffer pool. Possible values are **Default, Keep, Recycle** and not set (blank). The default is not set.

© Copyright, WhereScape Software Ltd All rights reserved.

Dimension Tables:

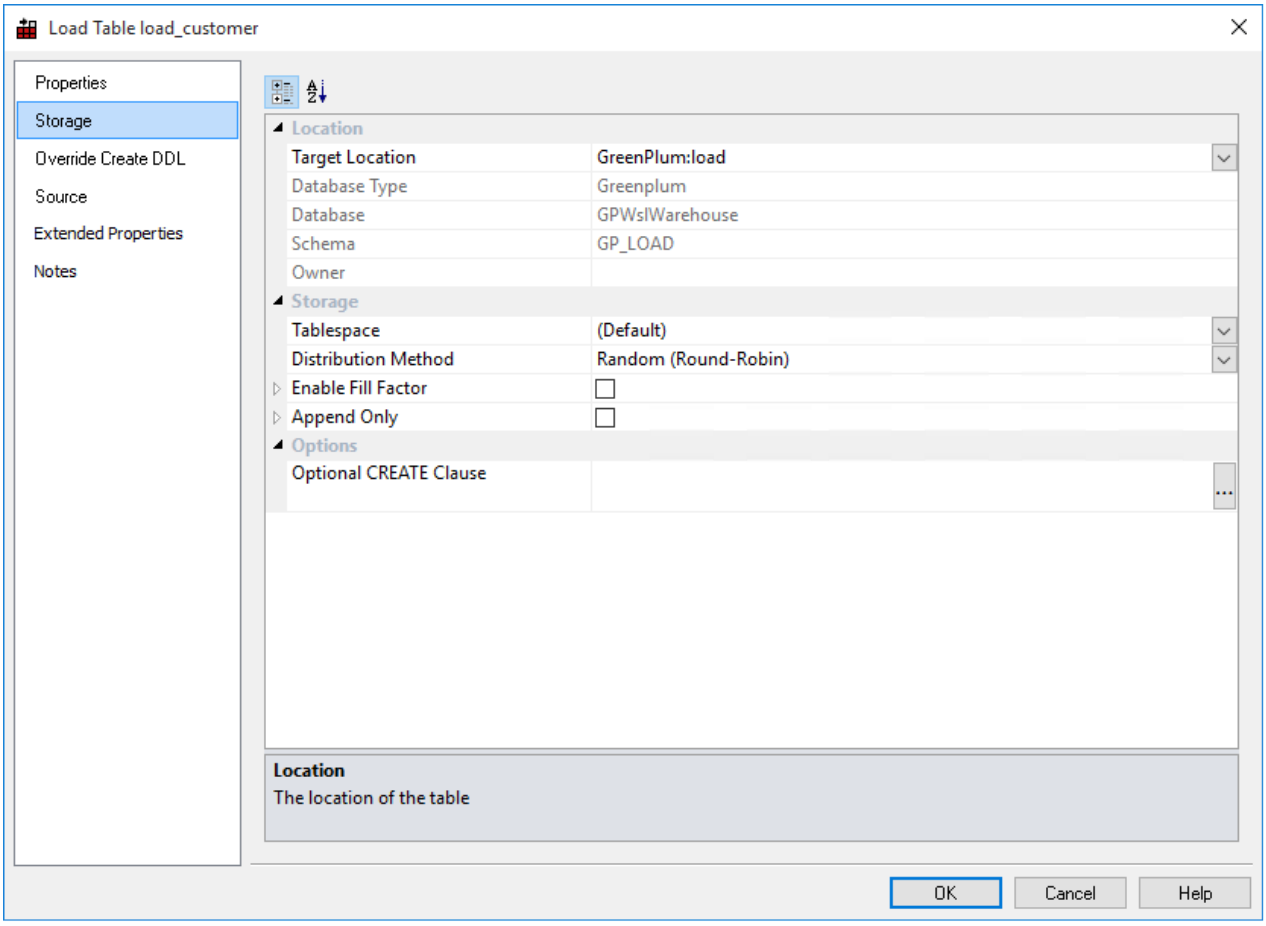


Fields	Description
Artificial Key Sequence Name	Name of the database sequence associated with the specific object. Enter the required new sequence name when creating a new table. The default key value for Dimensions is **dim_tablename_seq**. Specific Artificial Key Sequence Names can also be added to Data Store and EDW 3NF tables when an artificial key column is added to the table. **Warning** When altering the Artificial Key Sequence Name for existing tables, ensure the START WITH option is configured properly to avoid unique constraint violations.
Artificial Key Sequence Attributes	This option allows adding database specific and compliant options to the generated create sequence statement such as START WITH, INCREMENT BY or CACHE.

© Copyright, WhereScape Software Ltd All rights reserved.

Table Storage Screen - Greenplum

Typical **Storage** screen for a **Greenplum** table



Location

Fields	Description
Target Location	The target location that defines the path of the table. Select (local) for a local table or select the target schema if you are locating tables in different schemas **Note** The Force Target Usage setting in the **Tools > Options > Storage > Target Usage** removes the local target option from the drop-down list. Refer to **Target Usage** for details. • To add another database/schema to the list see more details on **Connections - Database or Connections - ODBC**. • To set default target locations for tables, refer to **Settings > Storage > Target Location**. **Note** When upgrading from a RED version previous to 6.8.2.0 and moving existing objects to a target location, all procedures that reference those objects need to be rebuilt. Any **FROM** clauses will also need to be manually regenerated in order for the table references to be updated to the new [TABLEOWNER] form.

© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
	Warning If the database type does not support moving tables, such as Oracle, Greenplum, Netezza, Teradata and PDW, all affected tables need to be **manually recreated** after any storage changes. Please note that changing the Storage for Dimension and Fact tables need to be handled very carefully, as artificial key relationships between Dimension and Fact could become out of sync. Recreating Fact tables and large Dimension tables might take a considerable amount of time.
Database Type	The database type for a connection that is used for target Datawarehouse tables.
Database	The database where the table is located. Leave blank to use the default for the connection or local environment.
Schema	The schema where the table is located.
Owner	Owner to which the object is altered.
Create DDL Template	Optional. Specify the template to use when creating a new DDL procedure script. This option is only visible if a DDL template is available for this database type. Default value is **None.**

Storage

Fields	Description
Tablespace	Default tablespace of the table that determines the storage location it is created in. Select (Default) to use the default.
Distribution Method	Select the Data Distribution Method. Options are Hash and Random (Round-Robin)

Enable Fill Factor

Tip

Setting this will enable setting of the Fill Factor.

Fields	Description
Fill Factor	Percentage [10-100] between 10 and 100. 100 (complete packing) is the default. When a small fill factor is specified, INSERT operations pack table pages only to the indicated percentage. The remaining space on each page is reserved for updating rows on that page.

Append Only

Tip

When set, will create the table as append-only. If this field is not set, the table will be created as a heap-storage table.

Fields	Description
Orientation	Columns selected to make up the Distribution Key.
Compression Type	Table-level compression is applied to an entire table.
Compression level	Table-level compression is applied to an entire table.

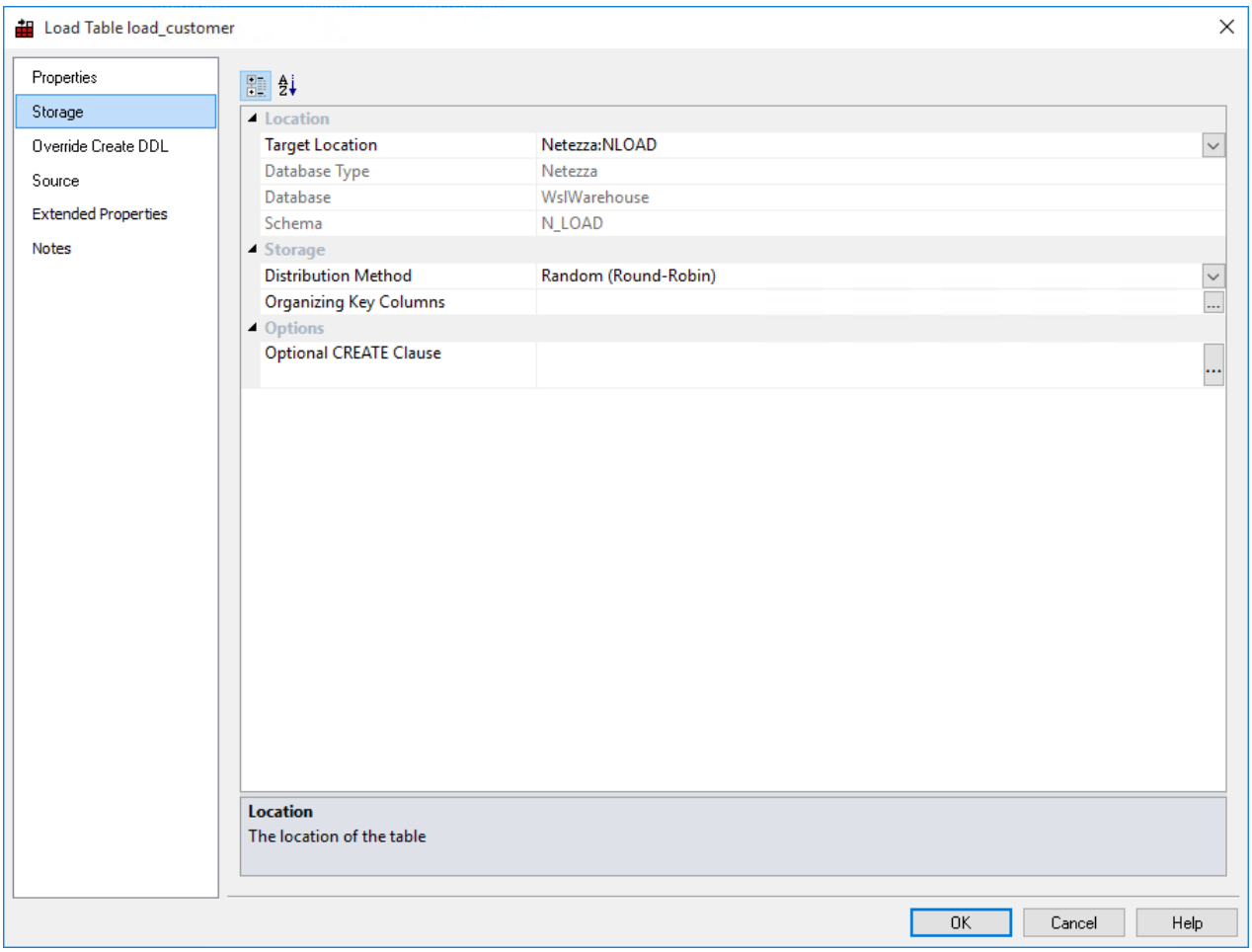
© Copyright, WhereScape Software Ltd All rights reserved.

Other

Fields	Description
Optional CREATE Clause	Database-specific-and-compliant DDL to append to the generated CREATE TABLE statement. **Tip** To set default create clause values for all new objects, go to the **Tools > Options > Storage** menu - **Default Optional CREATE Clause**.

Table Storage Screen - Netezza

Typical **Storage** screen for a **Netezza** table:



Location

Fields	Description
Target Location	The target location that defines the path of the table. Select (local) for a local table or select the target schema if you are locating tables in different schemas. **Note** The Force Target Usage setting in the **Tools > Options > Storage > Target Usage** removes the local target option from the drop-down list. Refer to **Target Usage** for details. • To add another database/schema to the list see more details on **Connections - Database or Connections - ODBC**. • To set default target locations for tables, refer to **Settings > Storage >Target Location**.

© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
	Note Target Location When upgrading from a RED version previous to 6.8.2.0 and moving existing objects to a target location, all procedures that reference those objects will need to be rebuilt. Any **FROM** clauses also need to be manually regenerated in order for the table references to be updated to the new [TABLEOWNER] form. **Warning** If the database type does not support moving tables, such as Oracle, Greenplum, Netezza, Teradata and PDW, all affected tables also need to be **manually recreated** after any storage changes. Please note that changing the Storage for Dimension and Fact tables need to be handled very carefully, as artificial key relationships between Dimension and Fact could become out of sync. Recreating Fact tables and large Dimension tables might take a considerable amount of time.
Database Type	The database type for a connection that is used for target Datawarehouse tables.
Database	The database where the table is located. Leave blank to use the default for the connection or local environment.
Schema	The schema where the table is located.
Owner	Owner to which the object is altered.
Create DDL Template	Optional. Specify the template to use when creating a new DDL procedure script. This option is only visible if a DDL template is available for this database type. Default value is **None.**

Storage

Fields	Description
Distribution Method	Select the Data Distribution Method. Options are Hash, Random (Round-Robin) and Netezza Designated.

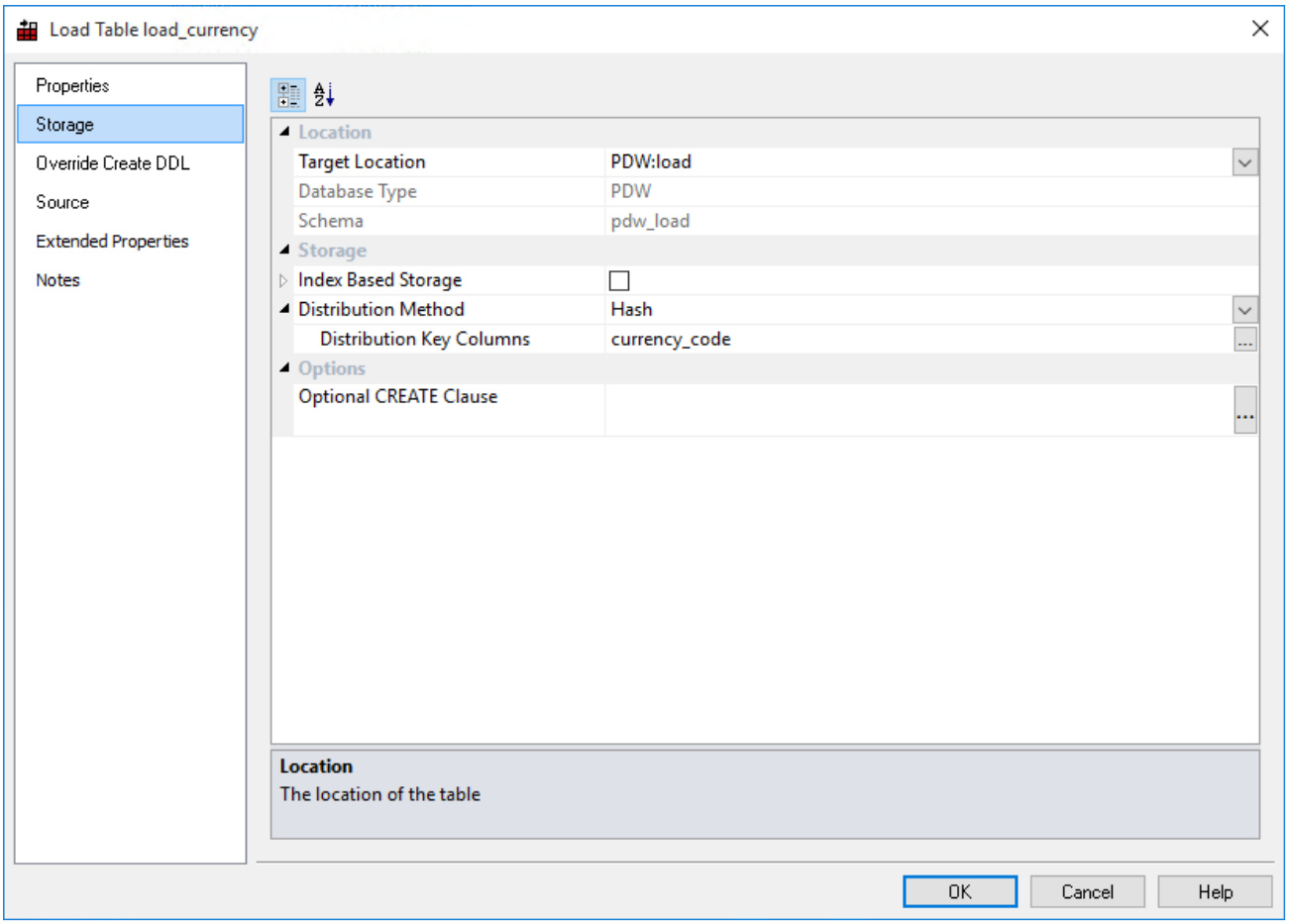
Other

Fields	Description
Optional CREATE Clause	Database-specific-and-compliant DDL to append to the generated CREATE TABLE statement. **Tip** To set default create clause values for all new objects, go to the **Tools > Options > Storage** menu - **Default Optional CREATE Clause**.

© Copyright, WhereScape Software Ltd All rights reserved.

Table Storage Screen - PDW

Typical **Storage** screen for a **PDW** table:



Location

Fields	Description
Target Location	The target location that defines the path of the table. Select (local) for a local table or select the target schema if you are locating tables in different schemas. **Note** The Force Target Usage setting in the **Tools > Options > Storage > Target Usage** removes the local target option from the drop-down list. Refer to **Target Usage** for details. • To add another database/schema to the list see more details on **Connections - Database or Connections - ODBC**. • To set default target locations for tables, refer to **Settings > Storage > Target Location**. **Note** When upgrading from a RED version previous to 6.8.2.0 and moving existing objects to a target location, all procedures that reference those objects need to be rebuilt. Any **FROM** clauses also need to be manually regenerated in order for the table references to be updated to the new [TABLEOWNER] form.

© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
	Warning If the database type does not support moving tables, such as Oracle, Greenplum, Netezza, Teradata and PDW, all affected tables also need to be **manually recreated** after any storage changes. Please note that changing the Storage for Dimension and Fact tables need to be handled very carefully, as artificial key relationships between Dimension and Fact could become out of sync. Recreating Fact tables and large Dimension tables might take a considerable amount of time.
Database Type	The database type for a connection that is used for target Datawarehouse tables.
Schema	The schema where the table is located.
Create DDL Template	Optional. Specify the template to use when creating a new DDL procedure script. This option is only visible if a DDL template is available for this database type. Default value is **None.**

Storage

Fields	Description
Index Based Storage	Allows setting of table data to be stored as either Clustered or Clustered Columnstore Index.
Index Type	Select the Index Type. Options are Clustered Columnstore Index or Clustered Index.
Distribution Method	Select the Data Distribution Method. Options are Hash and Random (Round-Robin)

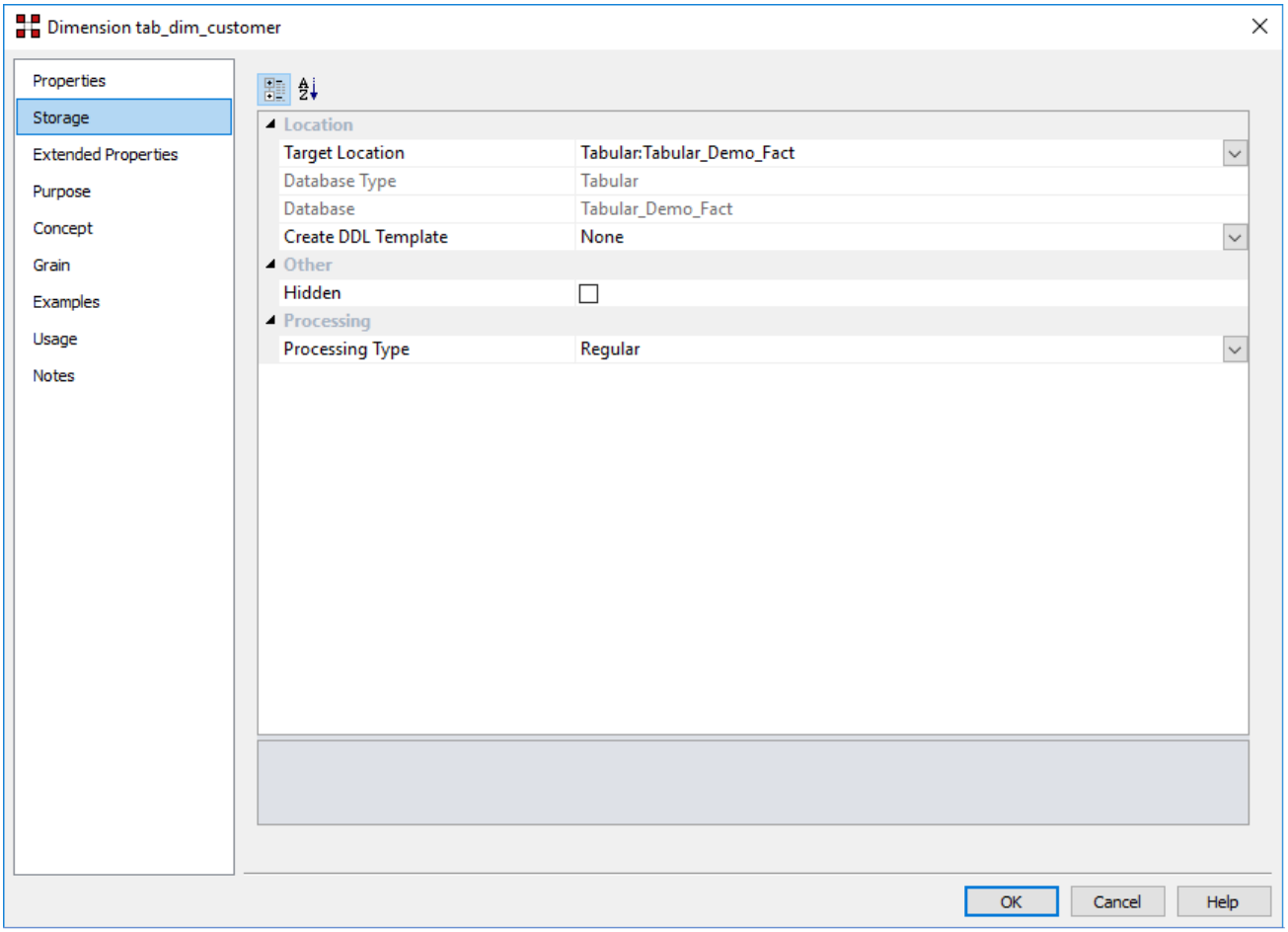
Other

Fields	Description
Optional CREATE Clause	Database-specific-and-compliant DDL to append to the generated CREATE TABLE statement. **Tip** To set default create clause values for all new objects, go to the **Tools > Options > Storage** menu - **Default Optional CREATE Clause**.

© Copyright, WhereScape Software Ltd All rights reserved.

Table Storage Screen - Tabular

Typical Storage screen for a **Tabular** table:



Location

Field	Description
Target Location	The path for the MSAS target. To add another MSAS target, refer to **Microsoft Analysis Server 2005+ - Tabular Mode** for details. To set default target locations for tables, refer to **Settings > Storage > Target Location**. **Note** The Force Target Usage setting in the **Tools > Options > Storage > Target Usage** removes the local target option from the drop-down list. Refer to **Target Usage** for details.
Database Type	For information purposes only, this displays Tabular, for MSAS Tabular targets.
Database	For information purposes only, this displays the database name.

Other

Field	Description
Hidden	Specifies whether the table is hidden from reporting client field lists.

Processing

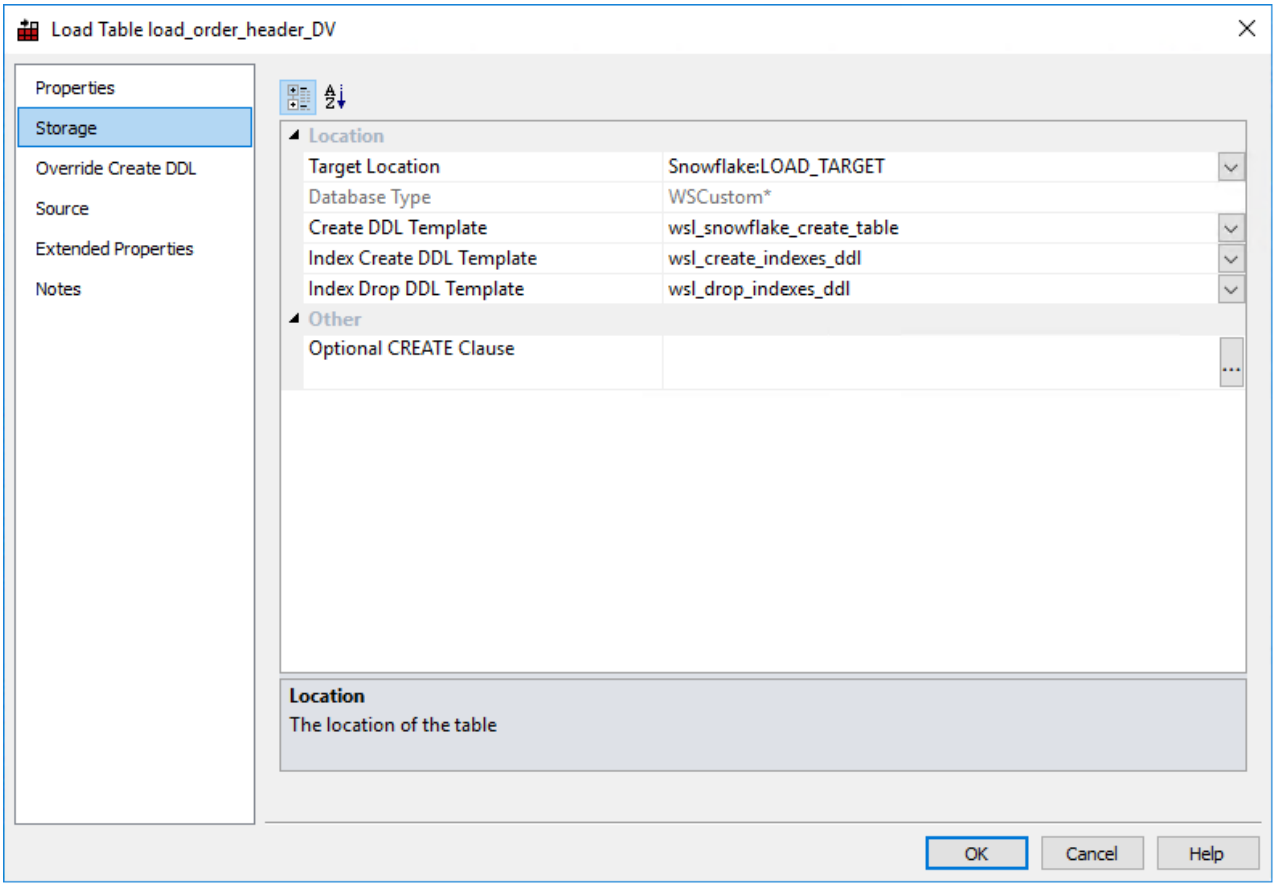
Field	Description
Processing Type	Set the database reporting tools processing to Regular or LazyAggregations in the

© Copyright, WhereScape Software Ltd All rights reserved.

Field	Description
	target Analysis Services database.

Table Storage Screen - Custom Database

Sample **Storage** screen for a **Custom Database** Table:



Location

Fields	Description
Target Location	The target location that defines the path to the location for the table.
Database Type	The database type for a connection that is used for target data warehouse tables.
Create DDL Template	Optional. Specify the template to use for generating DDL for this table. Set to *None* to use default DDL.
Index Create DDL Template	Optional. Specify the template to use for generating DDL to create Indexes on this table. Set to *None* to use default DDL.
Index Drop DDL Template	Optional. Specify the template to use for generating DDL to drop Indexes on this table. Set to *None* to use default DDL.

Other

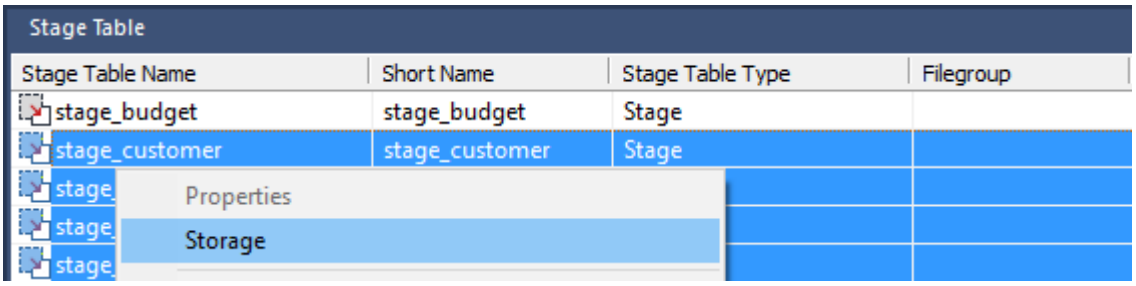
Fields	Description
Optional CREATE Clause	Database-specific-and-compliant DDL to append to the generated CREATE TABLE statement.

© Copyright, WhereScape Software Ltd All rights reserved.

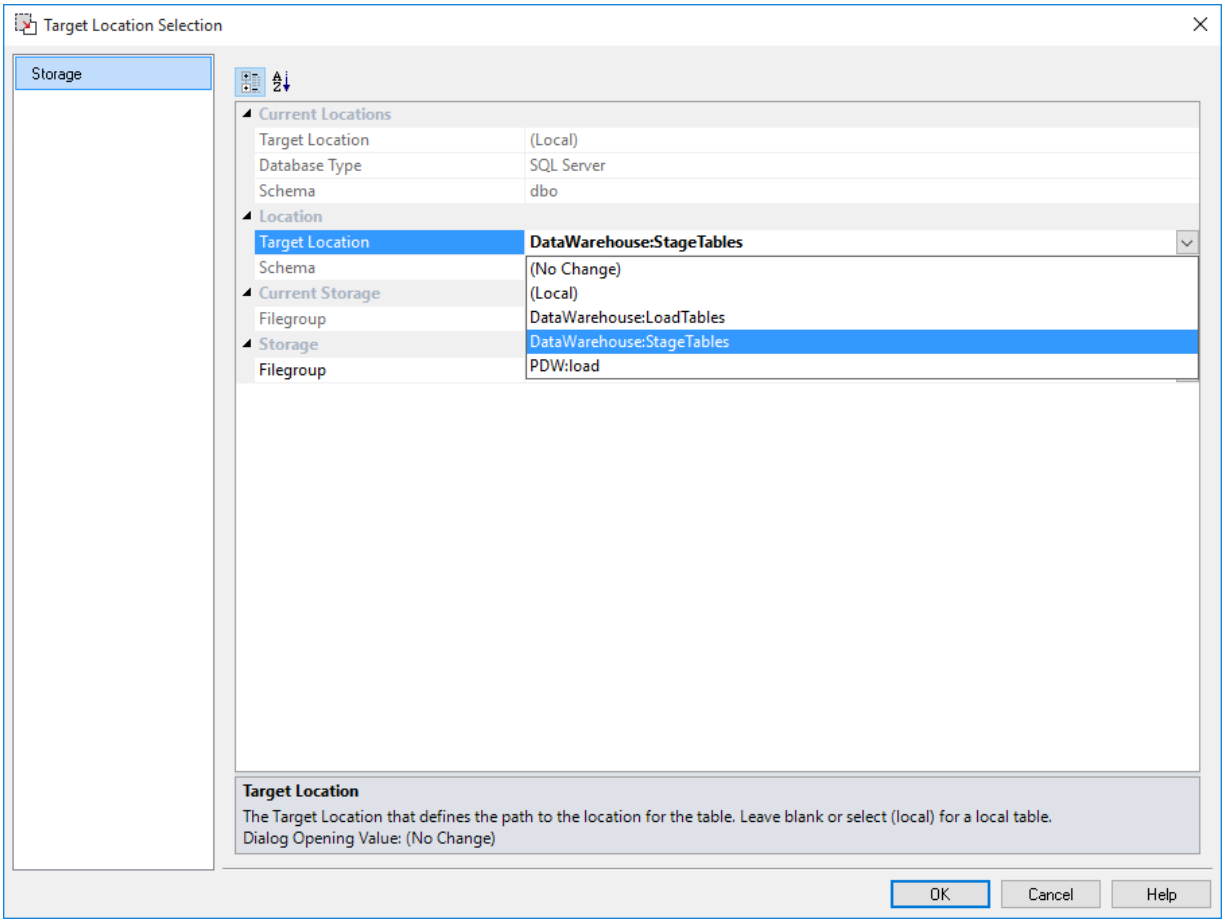
Bulk Table Storage Change

Table Storage locations can be changed through the Storage tab on a table's Properties dialog but they can also be changed in bulk by using the following process:

1. Double-click on the desired object group in the left pane. This will display all the tables in that group in the middle pane.
2. Select the tables that you wish to change the storage for using standard Windows selection.
3. Right-click to bring up a menu and select **Storage**.



4. On the Target Location Selection dialog, select the desired **Target Location** for the change.
   - Double-click on the desired object group in the left pane. This will display all the tables in that group in the middle pane.
   - Double-click on the desired object group in the left pane. This will display all the tables in that group in the middle pane.



Note

The Force Target Usage setting in the **Tools > Options > Storage > Target Usage** removes the local target option from the drop-down list. Refer to **Target Usage** for details.

5. Follow the next dialogs to complete the bulk storage change.

© Copyright, WhereScape Software Ltd All rights reserved.

6. Please note that **all procedures** from the affected tables will need to be **manually changed or regenerated** after a bulk storage change.

   If the database type does not support moving tables, such as **Oracle, Greenplum, Netezza, Teradata** and **PDW**, all affected tables will also need to be **manually recreated** after any storage changes.

   This also applies to moving SQL Server tables between databases, since SQL Server only support moving tables between schemas, within the same database.
   The tables moved are dropped from the source database but their metadata are stored in the destination database. The tables must be recreated in the destination database.

Warning

Please note that changing the Storage for Dimension and Fact tables need to be handled very carefully, because artificial key relationships between Dimension and Fact could become out of sync. Recreating Fact tables and large Dimension tables might take a considerable amount of time.

© Copyright, WhereScape Software Ltd All rights reserved.

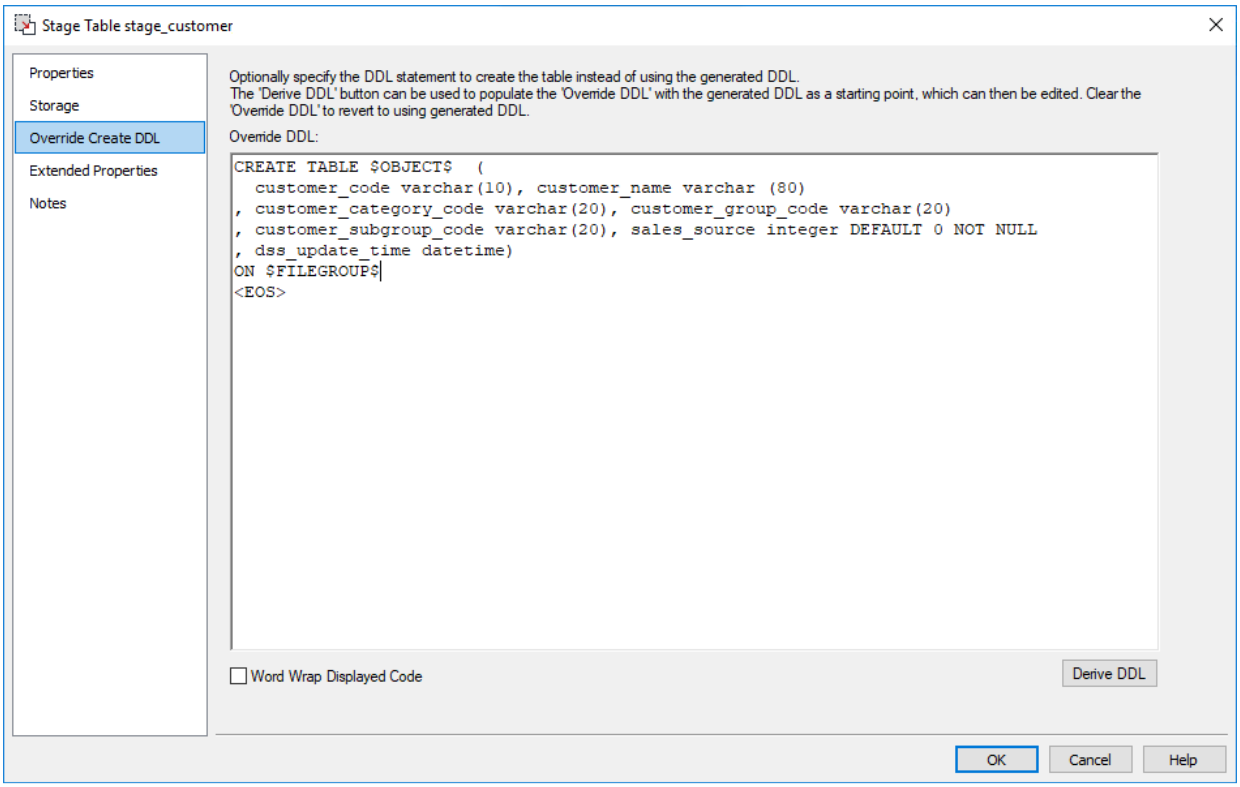
Override Create DDL

WhereScape RED enables you to create tables, using a specific DDL statement instead of the RED generated DDL.

You can override and edit the created DDL on load, stage, data store, EDW 3NF, dimension, fact, aggregate, user defined view and retro table's **Override Create DDL** tab.

Creating a table using a specific DDL statement:

1. If you are creating a **new** table, drag and drop the table from the right pane to the middle pane.
   - On the Properties dialog, click on the **Override Create DDL tab**.
   - Click on the **Derive DDL** button at the bottom of the dialog to get the generated DDL as your starting point.
   - Edit or clear the generated DDL and enter the desired DDL.
2. If you are **editing** an existing table, double click on the table object on the left pane to open the Properties dialog.
   - Click on the **Override Create DDL** tab.
   - Click on the **Derive DDL** button at the bottom of the dialog to get the generated DDL as your starting point.
   - Edit or clear the generated DDL and enter the desired DDL.



Note

The $OBJECT$ reference is auto translated to the fully qualified (schema/db.table name) at runtime.

- For SQL Server and Oracle respectively, $FILEGROUP$ and $TABLESPACE$ get replaced at runtime with the storage defined values.
- For DB2, $TABLESPACE$ and $INDEXSPACE$ get replaced at runtime with the storage defined values.
- For Greenplum, OWNER gets replaced at runtime with the storage defined values.

3. The "end of statement" indicator is <EOS> by default but can be configured in **Tools > Options > Code Generation>General**.

© Copyright, WhereScape Software Ltd All rights reserved.

Tip

To revert to RED deriving the original generated DDL at runtime, leave the Override DDL box blank or clear out the contents.
Clicking the **Derive DDL** button pops-up a warning message asking if you want to replace the current

Source

The **Source** tab is only available for Load tables. More information is available in the Loading Data chapter, or more specifically:

- **Database Link Load - Source Screen**
- **Loading data into RED load tables using SSIS**
- **Native ODBC Based Load - Source Screen**
- **Flat File Load - Source Screen**

Documentation Fields

Each table has a set of associated documentation fields. These are used to store descriptive metadata about the tables and how they are used.

The contents of each field appears in the generated documentation. Their order and placement can be customized in the generated documentation using the **Tools > Options** menu.

The following fields are available to store additional informational metadata that will appear in the generated WhereScape RED documentation:

- Purpose
- Concept
- Grain
- Examples
- Usage

Note

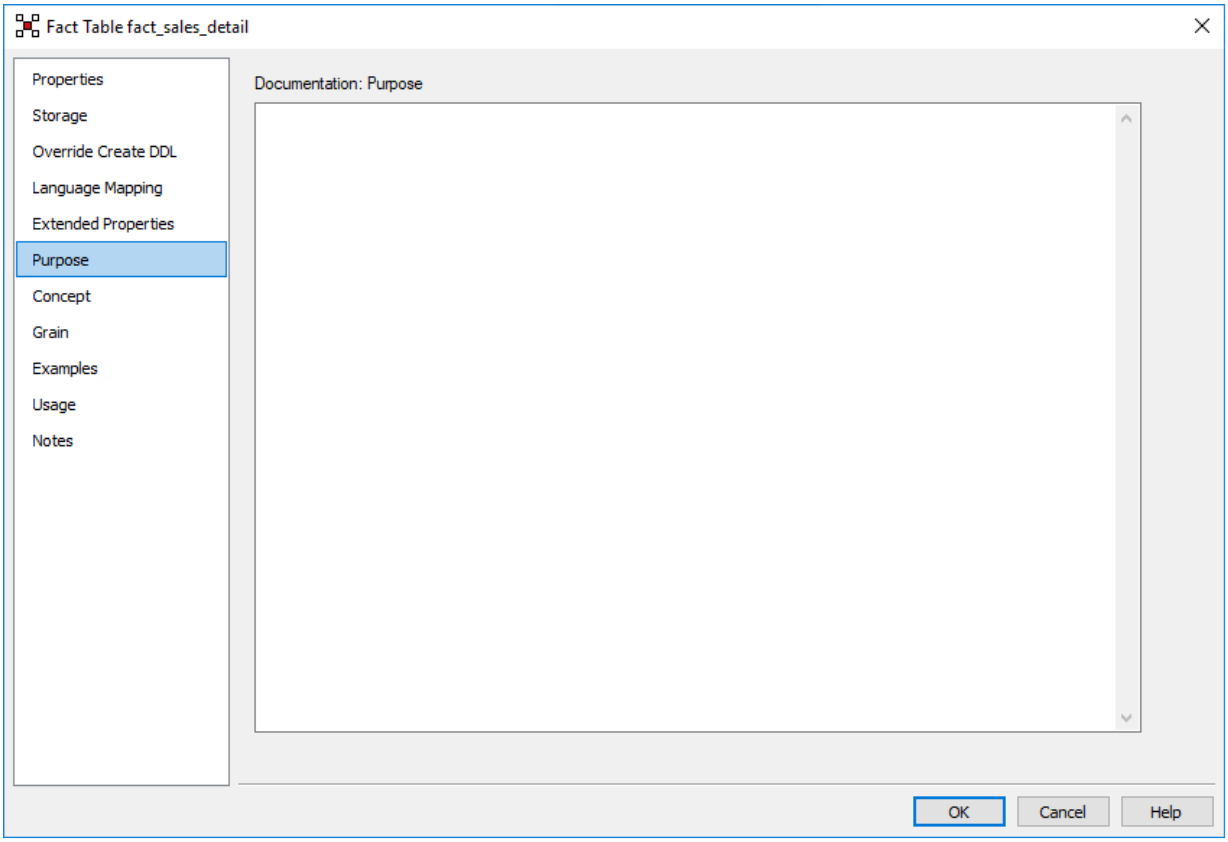
The names of these fields are completely arbitrary and can be changed in the generated documentation in the **Tools > Options > Documentation** menu.
By default, these fields are not enabled for load and stage tables, but can be enabled using the **Tools > Options > Documentation** menu.

Up to 4000 characters of information can be stored in each field.

Some or all of these fields can be removed from the documentation via the **Tools > Options > Documentation** menu.

© Copyright, WhereScape Software Ltd All rights reserved.

Documentation Fields Screen

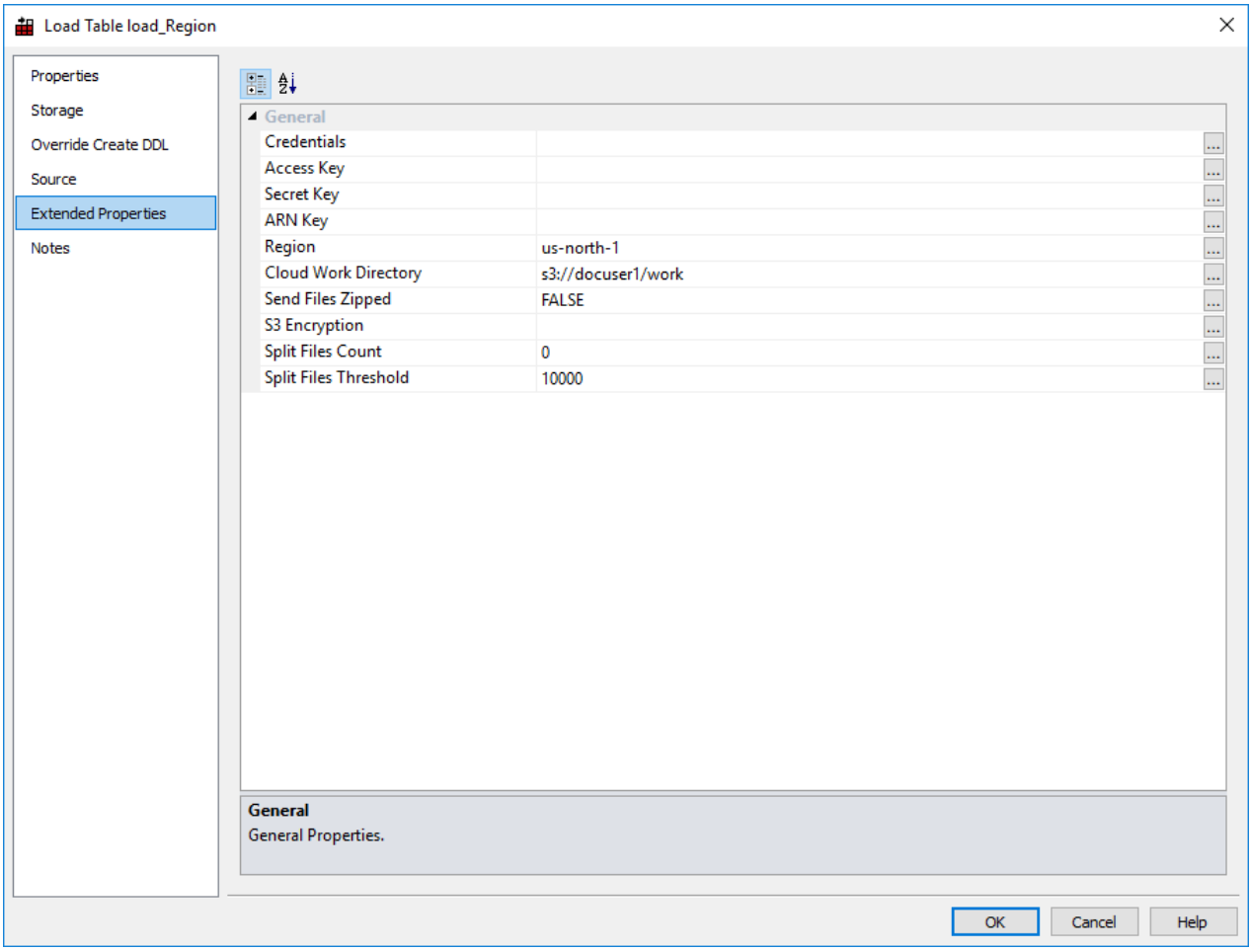


© Copyright, WhereScape Software Ltd All rights reserved.

Table Extended Properties

Note

These settings enable you to assign extended property values for a table object. Refer to **Extended Properties** for details.



© Copyright, WhereScape Software Ltd All rights reserved.

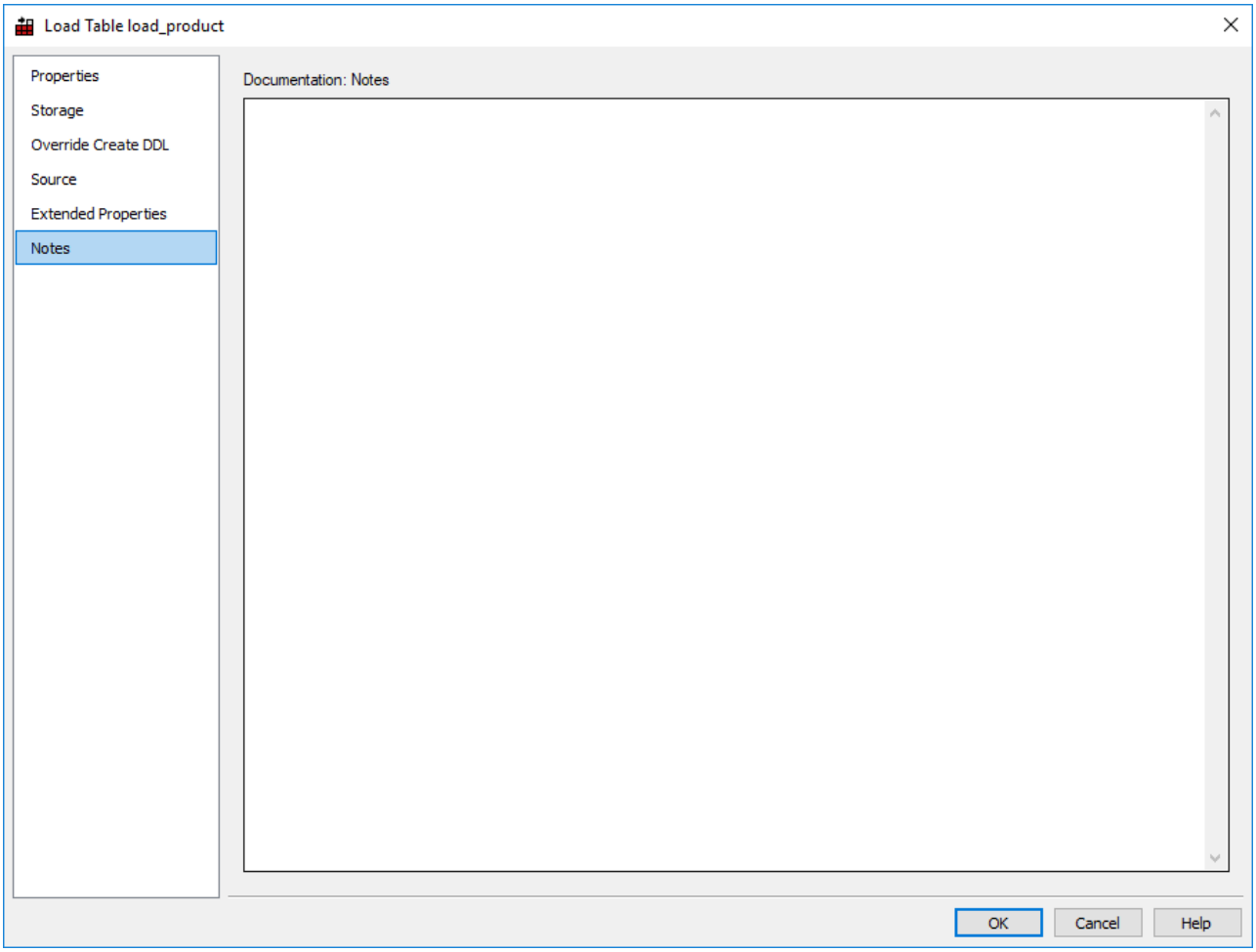
Notes

The **Notes** screen is used to enter notes for a particular WhereScape RED object.

The notes on an object are included in the WhereScape RED documentation. Up to 4000 characters of information can be stored in the Notes field.

By default, the Notes field is enabled for all object types.

The Notes field can be removed from the documentation via the **Tools > Options > Documentation** menu.



© Copyright, WhereScape Software Ltd All rights reserved.

Distributing Tables across Multiple SQL Server Databases

Note

This feature only applies to Microsoft SQL Server data warehouses.

The Multiple SQL database feature enables tables to be distributed across multiple databases, instead of only a single database. Table distribution enables different backup regimes, access security, or storage defaults, etc. to be implemented at an individual database level.

Previous versions of RED only allowed SQL Server target locations to be created on the RED Metadata connection. This resulted in the RED Metadata tables and all data warehouse tables being stored in the same database, as defined in this single connection.

WhereScape RED has enhanced the connection functionality to enable more than one SQL Connection, each including a combination of targets, a database attribute, and one or more schema, which provides the option of distributing the data warehouse objects across separate databases.

The following important points apply when using multiple databases:

- There can be one or more data warehouse database connections, all of which must be associated with a single SQL Server database server.
- Each database must have a unique ODBC DSN connection defined. The connection can have one or more targets defined, with each target having a database and a schema attribute.
- All source tables for OLAP Cubes and MSAS tabular database connections must be in the same database.
- The RED metadata tables must all reside in a single repository database.
- The associated update procedure of a table object residing in a data warehouse target (e.g. distinct schema/database in the same SQL Server instance) is stored in the same database as their target table by default. The location of stored procedures can be changed and locked via the **Procedure Properties** window. Refer to **Procedure Migration** for details.

Setting up Multiple SQL Databases

The following example illustrates the concepts and UI features that enable tables to be distributed across multiple databases.

The order presented below is the recommended procedural sequence.

In **Microsoft SQL Server Management Studio:**

Step 1: Create Databases and Schema

1. Create the required databases and schema and then for each database create a 32-bit ODBC DSN. The table below illustrates a basic example of multiple databases.

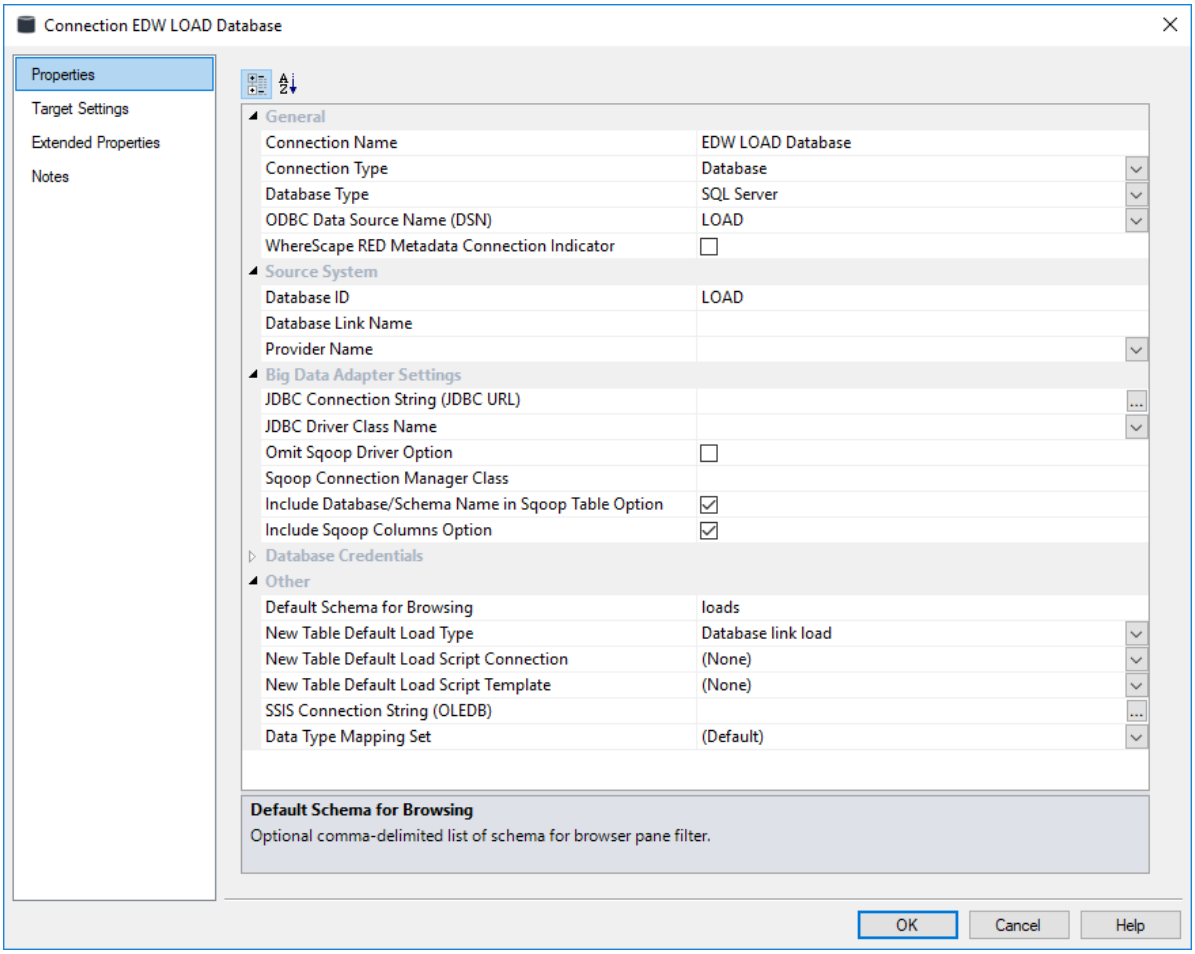
Database	Schema	ODBC DSN	Notes
WHERESCAPE	dbo	SQL Server Native Client… One DSN for each Database	RED Metadata Repository
LOAD	loads		For Load tables
STAGE	stages		For Stage tables
STARSCHEMA	dims facts		Two schema: • one for dimension tables • one for fact tables

In **WhereScape RED**:

© Copyright, WhereScape Software Ltd All rights reserved.

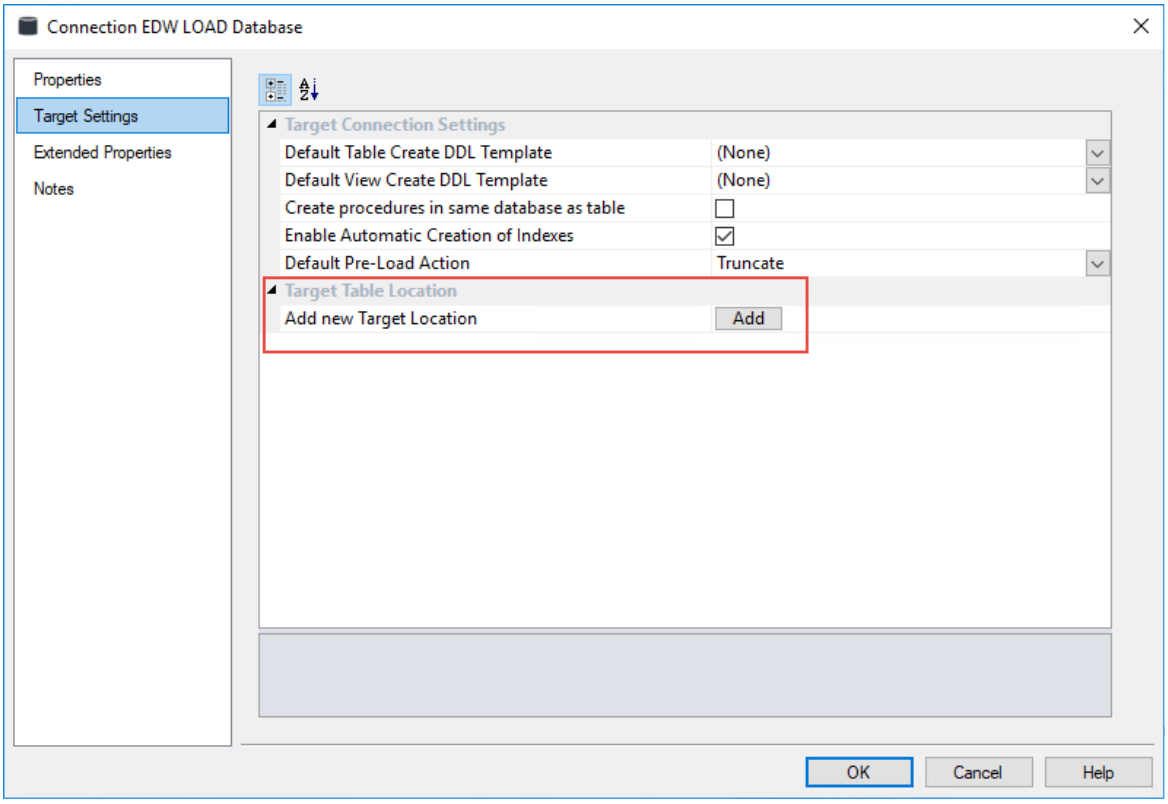
Step 2: Create Connections and Targets

1. Create a connection for each of the above databases and set the properties in the usual manner.
   - **Connection Type**: Database or ODBC
   - **Database Type**: SQL Server – the multi-database targets feature only applies to SQL Server
   - **ODBC DSN**: Specify the DSN that applies to the selected Connection
   - **Database ID**: Database Name - use the same name specified as the default database within the ODBC DSN. This database ID is used to auto-populate the database field in the target.



© Copyright, WhereScape Software Ltd All rights reserved.

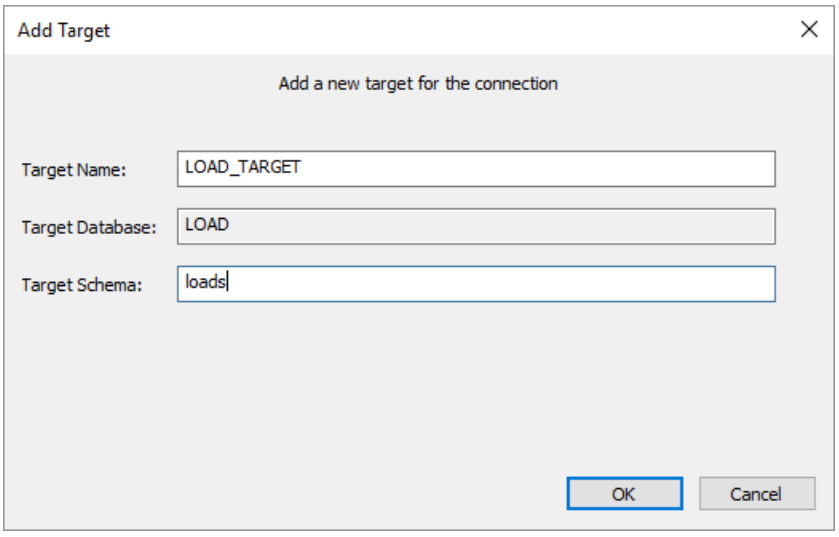
2. The **Add new Target Table Location** in the **Target Settings** tab is used to specify the target for each schema within the target database connection.



Notes

The **Create procedures in same database as table** option enables you to specify if the procedures associated with new Tables created in Targets of the connection is created in the same database as the Table objects, or in the RED repository meta database.
This option is not selected (OFF) by default, e.g. the Table's stored procedures are created in the RED repository meta database.
If the Table is created in a different Target location (e.g. distinct schema/database in the same SQL Server instance), you can turn ON this option, if you want the stored procedures to also be located in the same Target database.
The location of stored procedures can be changed and locked via the **Procedure Properties** window.

Add one or more targets for the connection, specifying a **Target Name** and the **schema** to use.

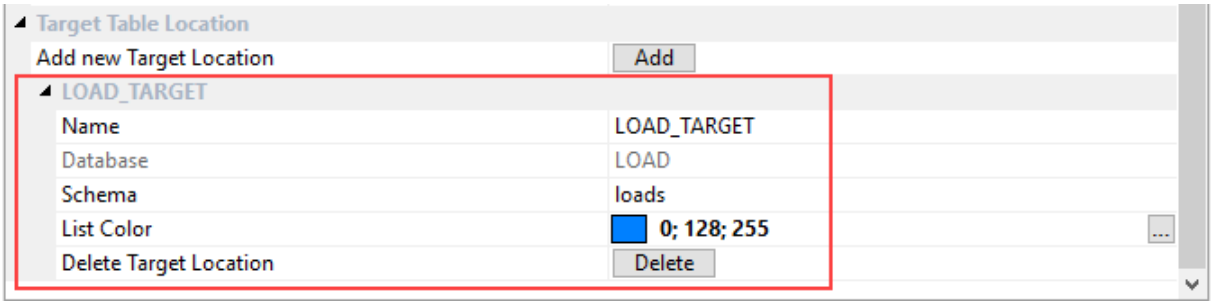


© Copyright, WhereScape Software Ltd All rights reserved.

Note

The Target Database is auto-populated, using the database ID specified in the connection.

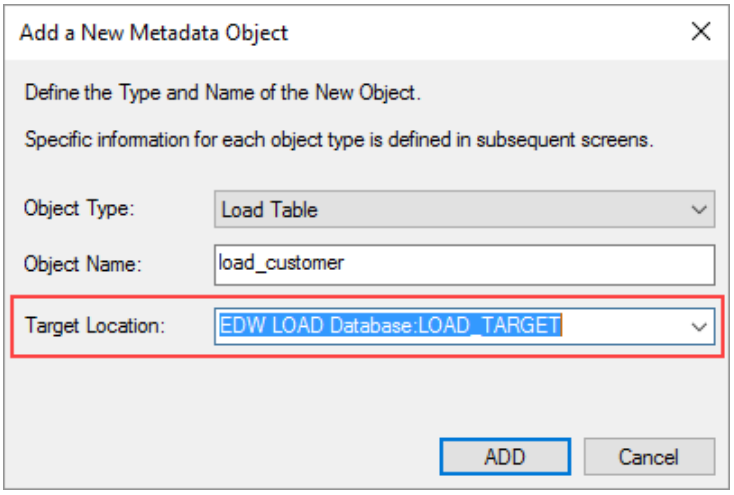
The list color can be optionally changed as required after the target table has been created.



3. Repeat actions 1 and 2 of Step 2 for each database and its schema.

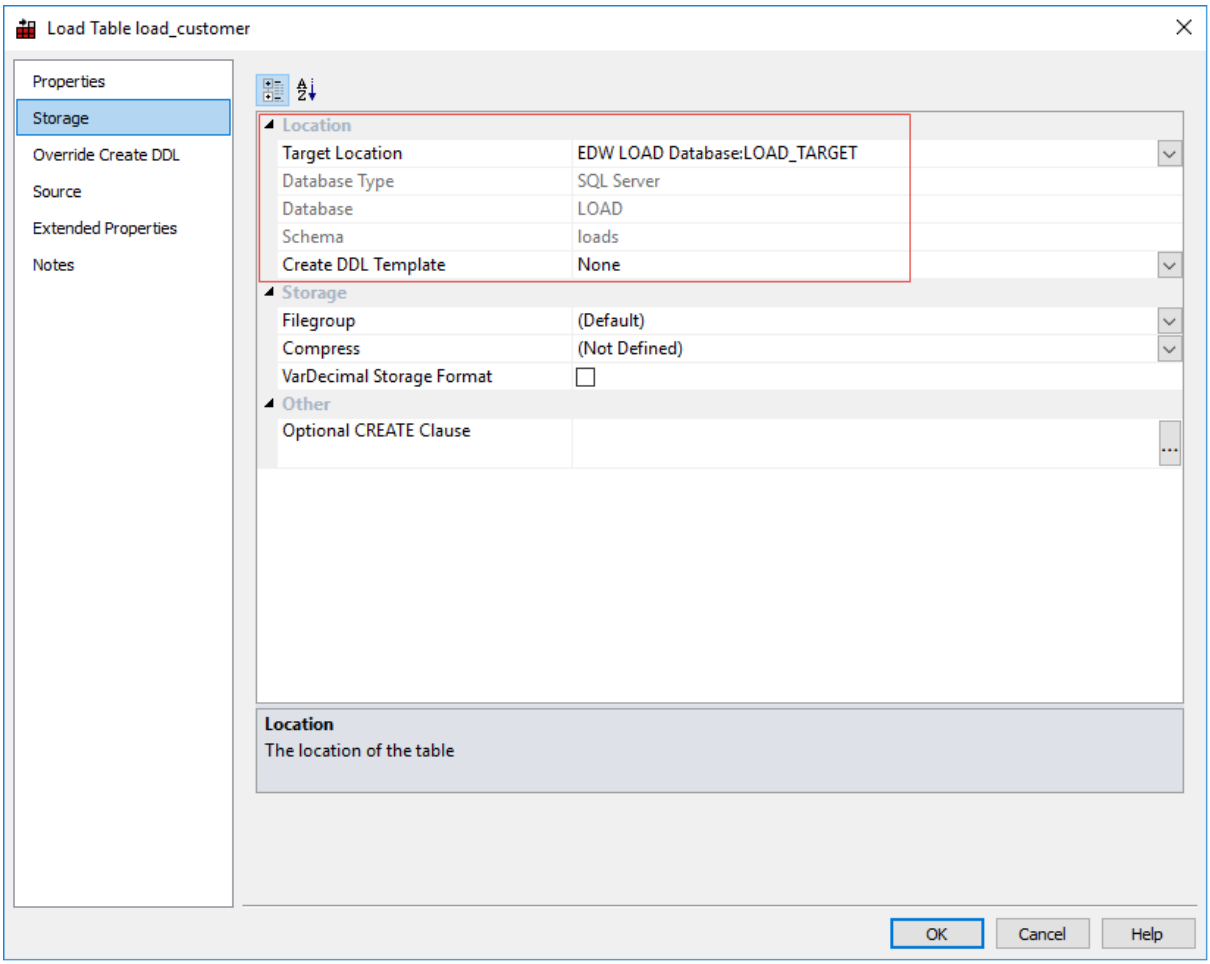
Step 3: Create New Objects

1. Once the connections and targets have been created, **Add** new objects in the usual manner. However, when using multiple databases it is important to select the correct database in which to store the object.



© Copyright, WhereScape Software Ltd All rights reserved.

The **Storage Properties** screen shows the storage location details for the selected table.

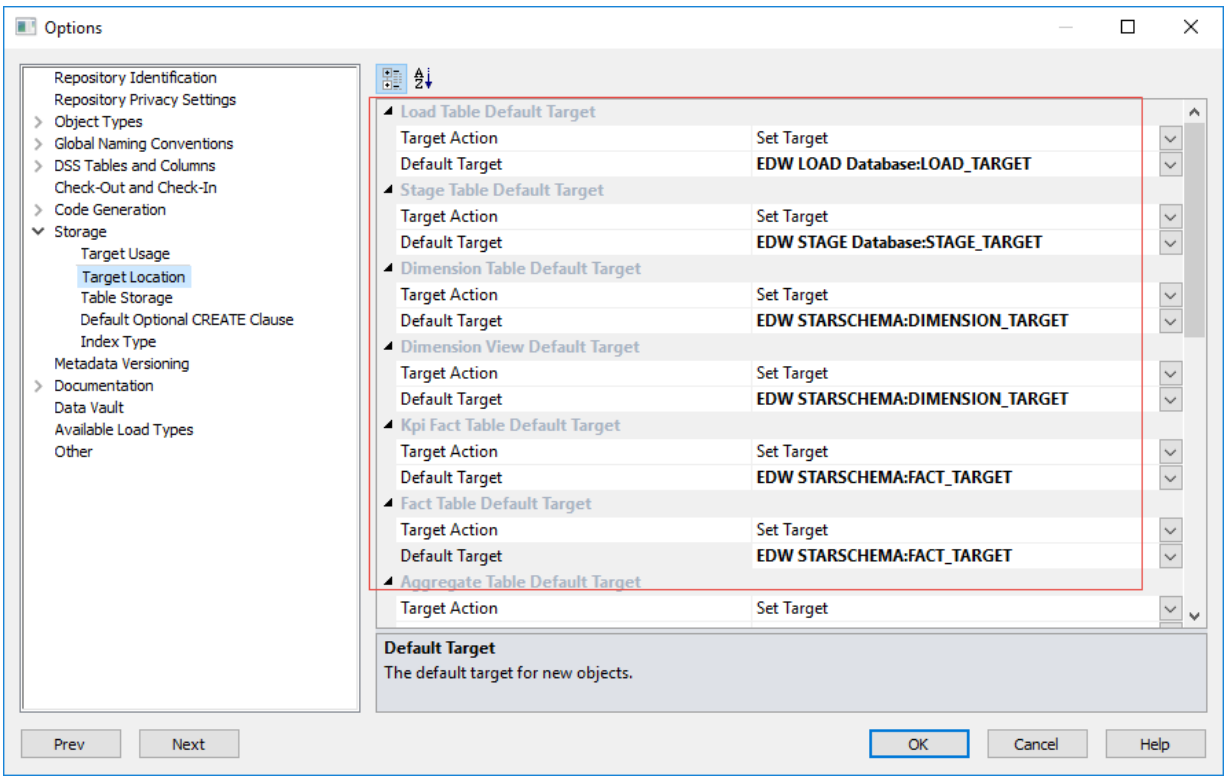


© Copyright, WhereScape Software Ltd All rights reserved.

Step 4: Optional Default Target

Default Target Locations can be optionally specified in **Options > Storage > Target Location**, e.g. all new Load tables are created in a specific target database/schema you defined. This saves time when creating new tables.

1. Open the drop down list for the required target and select **Set Target**.
2. Open the drop down list and select the required **database target** to be used as the default.
3. Click **OK** when all the changes are complete.



© Copyright, WhereScape Software Ltd All rights reserved.

Loading Data

Load tables are the first entry point for all information coming into the data warehouse. There are multiple ways of getting data into Load tables.

Load Type	Description
Database Link Load	The data is loaded from another database
External Load	The Load table is populated by some external process, e.g. an ETL (Extract, Transform and Load) tool or EAI (Enterprise Application Integration) tool, putting the data directly into the Load tables.
ODBC Load	The data is loaded via an ODBC connection, either directly by reading and inserting each row (a Load Type of **ODBC load**), or via a file by reading from the source system using ODBC and writing to a file and then loading the file (a Load Type of **Native ODBC**). ODBC connections require a Windows scheduler.
File Load	A flat file load where most of the processing is managed and controlled by the scheduler.
Script-based load	A flat file load where a host system, e.g. UNIX or Windows script file is executed to perform the load. Script-based loads on Windows supports both DOS Batch and **PowerShell scripts**.
XML File load	This method loads an XML file from a Windows directory.

Notes:

1. A **Windows scheduler** must be available to handle ODBC based loads.
2. The following load types are *not* enabled by default for newly created Teradata repositories:
   - Integration Services (SSIS) load in Database, ODBC and Windows connections.
   - Native ODBC load in an ODBC connection.
   - XML file load in a Windows connection.

   Refer to **Tools > Options > Available Load Types** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

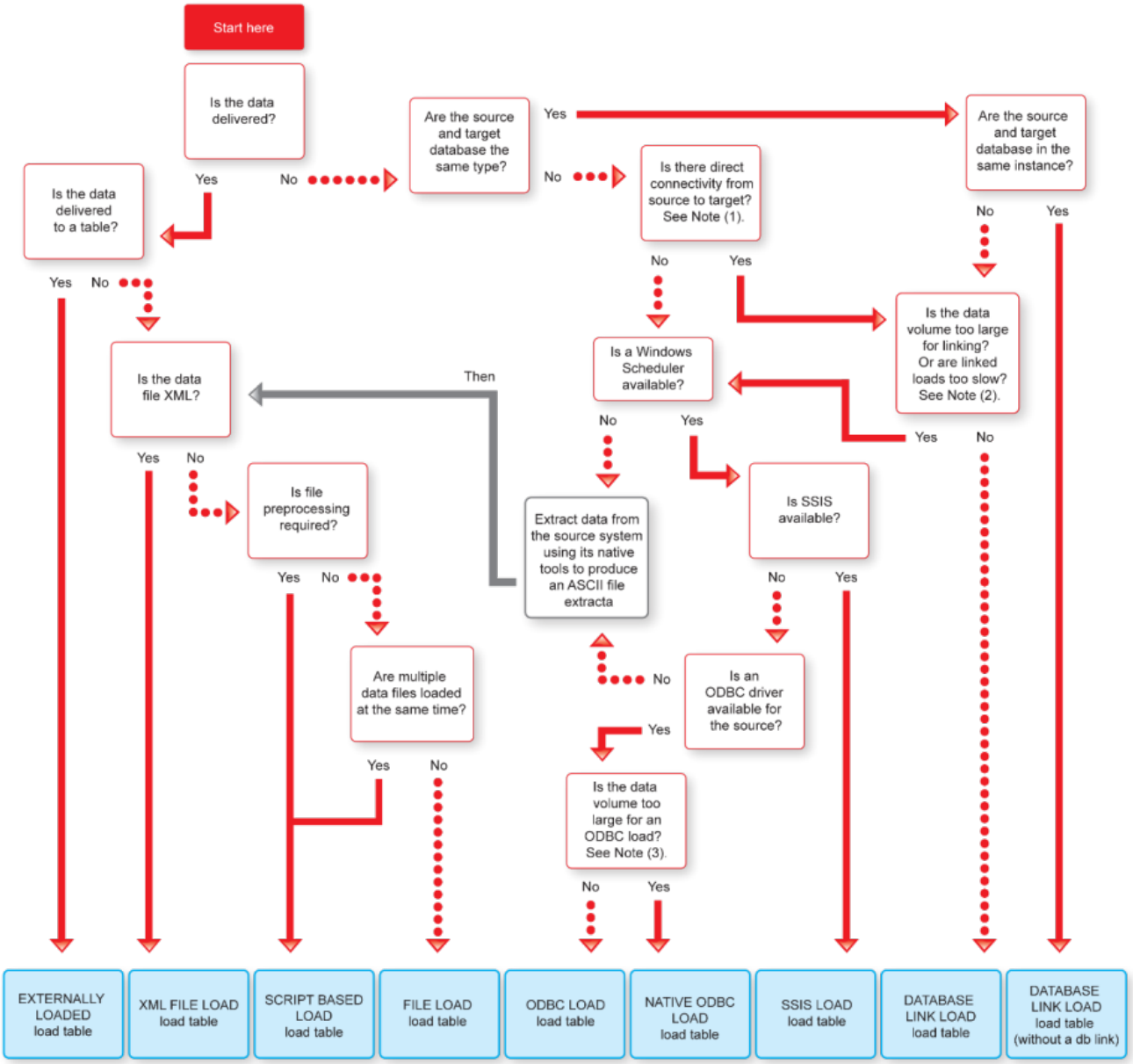
Choosing the Best Load Method

Several different factors need to be considered when choosing the best type of load table to use:

- Source and target database types
- Locations of source and target databases
- Available connectivity options and performance
- Amount of data being loaded
- Is the data delivered or fetched?
- For delivered data, what format is it in and does it require processing to make it loadable?

Load Table Decision Tree

The following decision tree can be used when selecting the best type of load table:



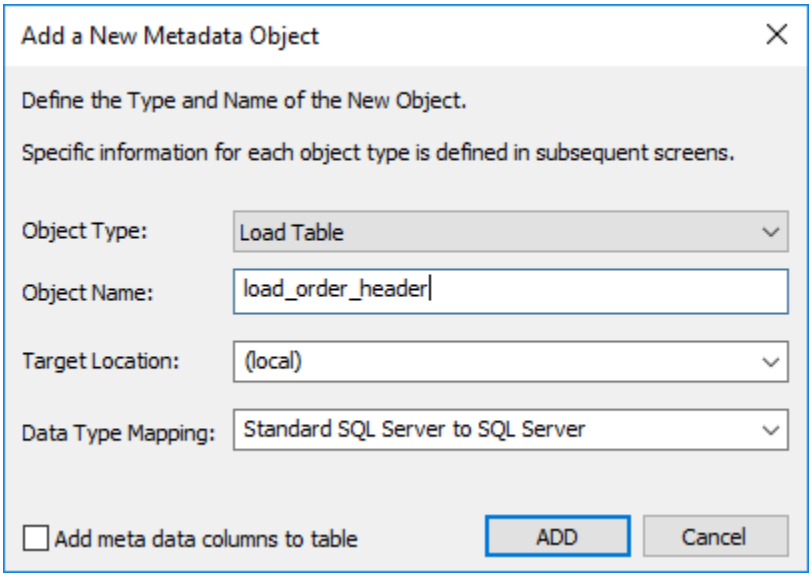
© Copyright, WhereScape Software Ltd All rights reserved.

Load Drag and Drop

The simplest way to create a Load table is to use the drag and drop functionality of WhereScape RED.

Drag and drop can be used for all connection types and the process is the same in all cases.

1. Browse to the source system connection (**Browse > Source Tables**).
2. Create a drop target by double-clicking the Load Table object group in the left pane. The middle pane should have a column heading of **Load Table Name** for the leftmost column.
3. Select a table or file in the right pane and drag it to the middle pane. Drop the table or file anywhere in the middle pane.
4. Answer the prompts to create the Load table.
5. When using targets, you can also change the predefined **Target Location** settings in **Tools > Options > Storage**. At the time of the table object's drag and drop, the **Add a New Metadata Object** window enables you to edit the **Target Location** and **Data Type Mapping**, so the table object's location can be changed on a table by table basis.
   The **Data Type Mapping** field is automatically set, based on the source and the **Target Location** selected, but can be changed if needed.



Note

The option **Add meta data columns to table** is selected when creating Load tables that are used in creating Data Vault objects. If this option is selected, two DSS columns (**dss_record_source** and **dss_load_date**) are included in the meta data for the table and are populated by transformations. These two DSS columns could equally be applied to other Load tables not used in a Data Vault system but are particularly important to comply with the Data Vault standards. Refer to **Data Vaults** for details.

WhereScape RED supports loading tables of up to 2048 columns. However, this maximum number of column loading restriction can in fact be lower than the target database or tools permit.

The target database will provide users the appropriate warning, if the maximum number of columns loaded is breached at runtime.

Notes

When creating a Load table in WhereScape RED by dragging and dropping a source table, RED reads the structure of the table in the source system and attempts to construct an equivalent Load table in the data warehouse. There are occasions when the Load table creation will fail due to incompatible data types in the target data warehouse. The remedy is to change the data types of the particular attributes which are causing the load failure. Once corrected, the Load table is created.
It is important that the table load is tested to ensure that data can be INSERTED into the Load table from the source table. If the load fails then the data may need to be explicitly converted to the destination data

© Copyright, WhereScape Software Ltd All rights reserved.

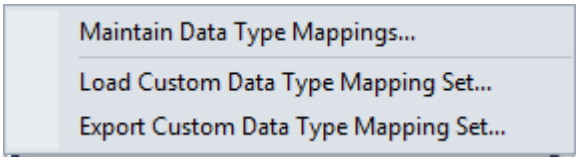
Notes

type, using a column transformation that is executed during the load (refer to **Load Table Column Transformations** for details).
Incompatible data types that cause Load table creation errors are typically caused by:

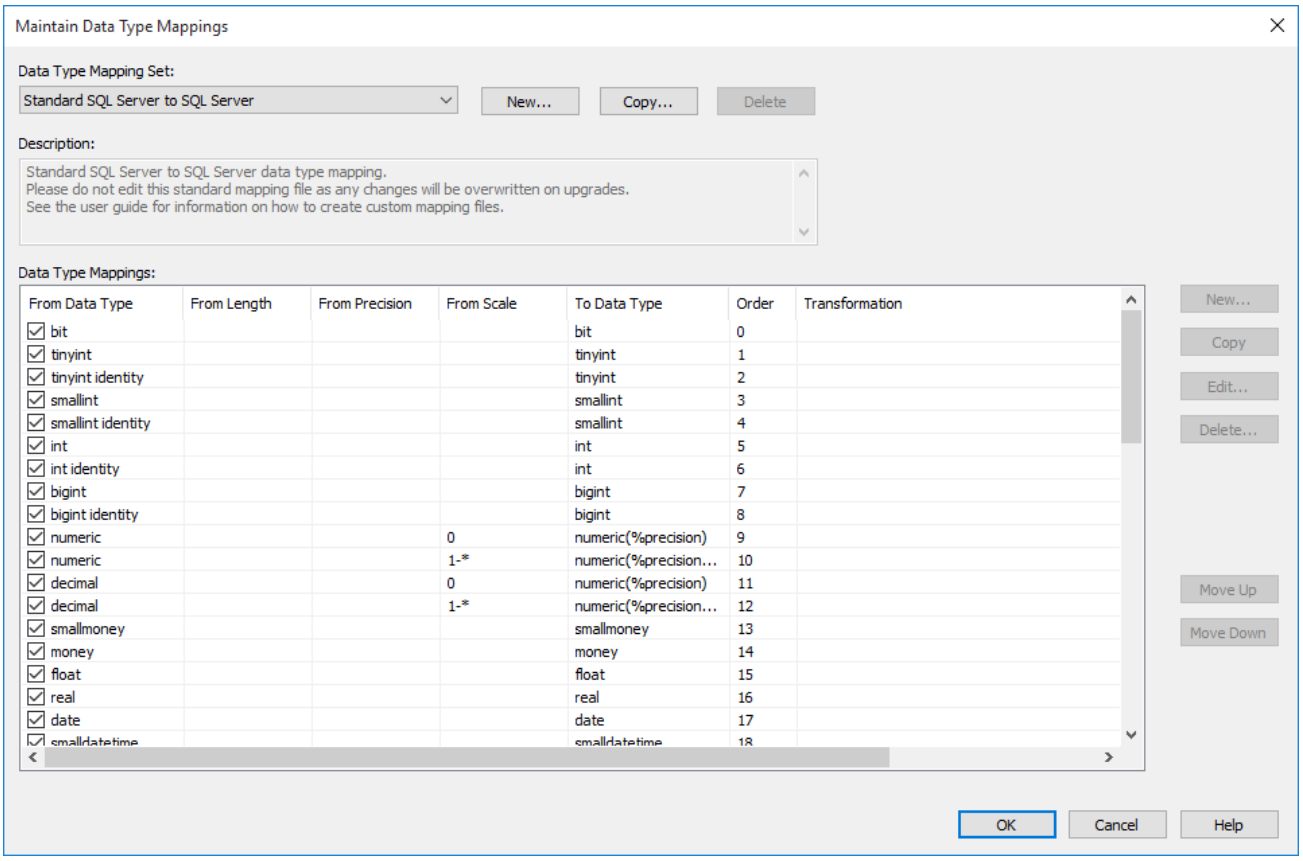
1. User defined data types in the source database.
2. Incorrect data type mapping during Load table definition in WhereScape RED.
3. Data types that cannot be inserted into (e.g. identity columns in SQL Server).

Data Type Mappings

In **Tools > Data Type Mappings**, the first menu option enables you to maintain the data type mappings.



The **Maintain Data Mappings** window is as follows:



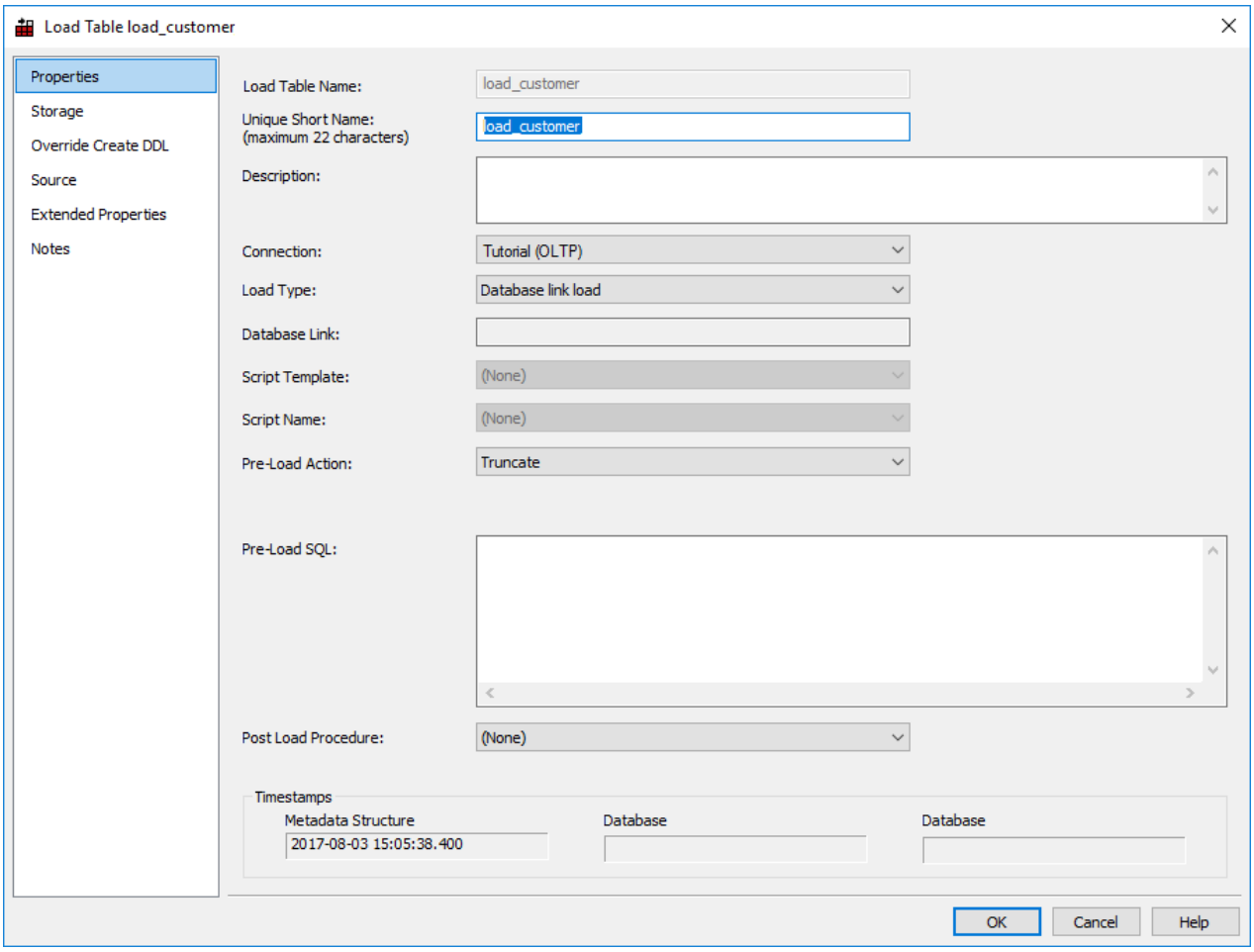
Refer to **Data Type Mappings** for details about how to manage data mappings in RED.

© Copyright, WhereScape Software Ltd All rights reserved.

Database Link Load

Database Link Load - Properties

The fields in the Database Link Load Properties screen are described below:



Fields	Description
Load Table Name	The table name is limited by most relational databases to a maximum of 30 characters and must be unique. Table name defaults can be set up in **Tools > Options** to define a prefix or a post fix that can be added in order to identify clearly that this is a load table. Example: **load_**customer. By default, WhereScape RED uses the prefix **load_** for Load tables.
Unique Short Name	The table short name is limited in size to 22 characters in Oracle and SQL Server and 12 characters in DB2. It must be unique. The short name is used in the naming of indexes, keys, procedures and scripts.
Description	Enter a description of the table. This description appears in the documentation that can be generated, once the data warehouse is built.
Connection	Enter the connection being used to get the data. The connections for Load tables can be changed in bulk, refer to **Changing load Connection and Schema** for details.
Load Type	The load type is typically defined by the connection, and should not normally be changed. This drop-down shows all valid load types for the connection.
Database Link	The database link is displayed in this field.
Script Template	The script template used for a script based load.
Script Name	This field is only active for script based loads.

© Copyright, WhereScape Software Ltd All rights reserved.

<table>
<tr><th>Fields</th><th>Description</th></tr>
<tr><td></td><td>Tip
When doing a Script based load, use the Rebuild button after selecting the relevant script to be rebuilt from the Script Name drop-down list.</td></tr>
<tr><td>Pre-load Action</td><td>Select an action to be performed on the table before the load occurs. Options are:
Truncate
Execute Pre-Load SQL
Both Truncate and Execute Pre-Load SQL
No action</td></tr>
<tr><td>Pre-load Sql</td><td>If a Pre-load Action of Execute pre-load Sql was selected, then the SQL statement to execute before the load takes place should be entered in this box.</td></tr>
<tr><td>Truncate Options</td><td>Oracle Only: WhereScape RED enables controlling the Oracle TRUNCATE options, such as REUSE/DROP STORAGE for load processing.
Add a Truncate option for Load table processing by typing it in the Truncate options box in the Load table Properties screen.
Truncate options can also be added to the procedure generation in the Define Procedure Type window.
The contents of Pre-Load SQL can be an SQL statement or a procedural block.
If using a single statement then the trailing semi-colon is not required for Oracle.
If using a procedural block then the final semi-colon is required. The following examples illustrate the possible values in this field. Note the trailing semi-colon on the procedural block example.
<table><tr><th>Example of pre-load statements:</th></tr><tr><td>delete from load_customer where code < 23</td></tr><tr><td>truncate table load_customer</td></tr><tr><td>DECLARE

v_status number;

v_result varchar2(256);

BEGIN

update_dim_customer(1,'a','b',0,0,v_result, v_status);

END;</td></tr></table>Note:
Not all pre load SQL commands works in a SQL Server database. These commands are executed as a batch. For example: create trigger as the second command will fail because it is not supported in batch mode, unless it is the first command.</td></tr>
<tr><td>Table Properties clauses (e.g. Partition)</td><td>These are clauses that are added to the end of the table create statement. Typically used for putting partition information on the table. In Oracle, this can also be used to add a parallel clause or compression clause to the table.</td></tr>
</table>

© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
Post Load Type	**Note:** This field is only available for custom database targets. Enables you to specify the type of post load to execute for the table. Options are **Procedure** or **Script**.
Post Load Procedure	A procedure that is executed immediately following the load. If you execute an externally loaded table, no load occurs, but a post load procedure can still be executed. Post load procedures can either be manually generated from a RED provided procedure outline or generated using a RED template—refer to **Rebuilding Update Procedures** for details.
Post Load Script	**Note:** This field is only available for custom database targets and is only displayed when the **Script** option is selected from the **Post Load Type** field above. A script that is executed immediately following the load. If you execute an externally loaded table, no load occurs but a post load script can still be executed. Post load scripts can be generated using a RED template—refer to **Rebuilding Update Procedures** for details.

Note:

Three fields at the bottom of the Load Table Properties screen display date information:

1. Date table structure last updated.
2. Date created in database.
3. Date last updated in database.

Database Link Load - Source Mapping

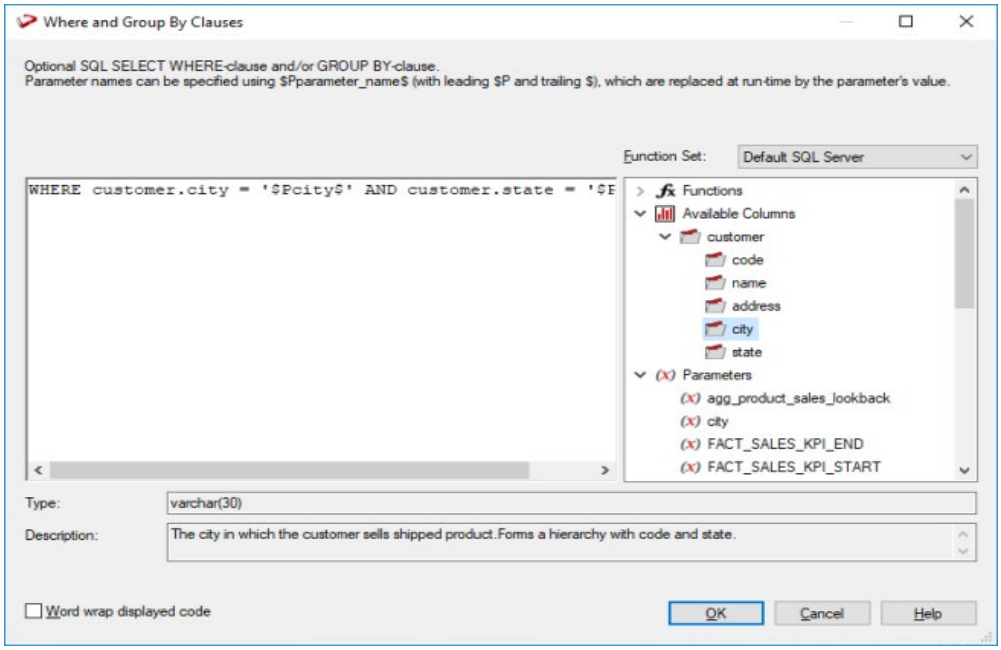
The fields in the **Database Link Load Source Screen** are described below:

Fields	Description
Load Type	Method of loading data into the table. The available options are dependent on the **Source Connection**. Defaults to the **Default Load Type** of the **Source Connection**. Can be specified via the table **Properties** window.
Source Connection	The connection that identifies the source database. Can be specified via the table **Properties** window.

General

Fields	Description
Select Distinct Values	Include the DISTINCT keyword in the SQL SELECT statement.
Source Schema	Schema within the source database where the source table resides.
Derive Source Tables(s) and Source Columns	Derive the Source Table(s) and Source Column(s) properties (of this screen) from the source details of this table's columns. **Note** The existing property values will be overwritten.

© Copyright, WhereScape Software Ltd All rights reserved.

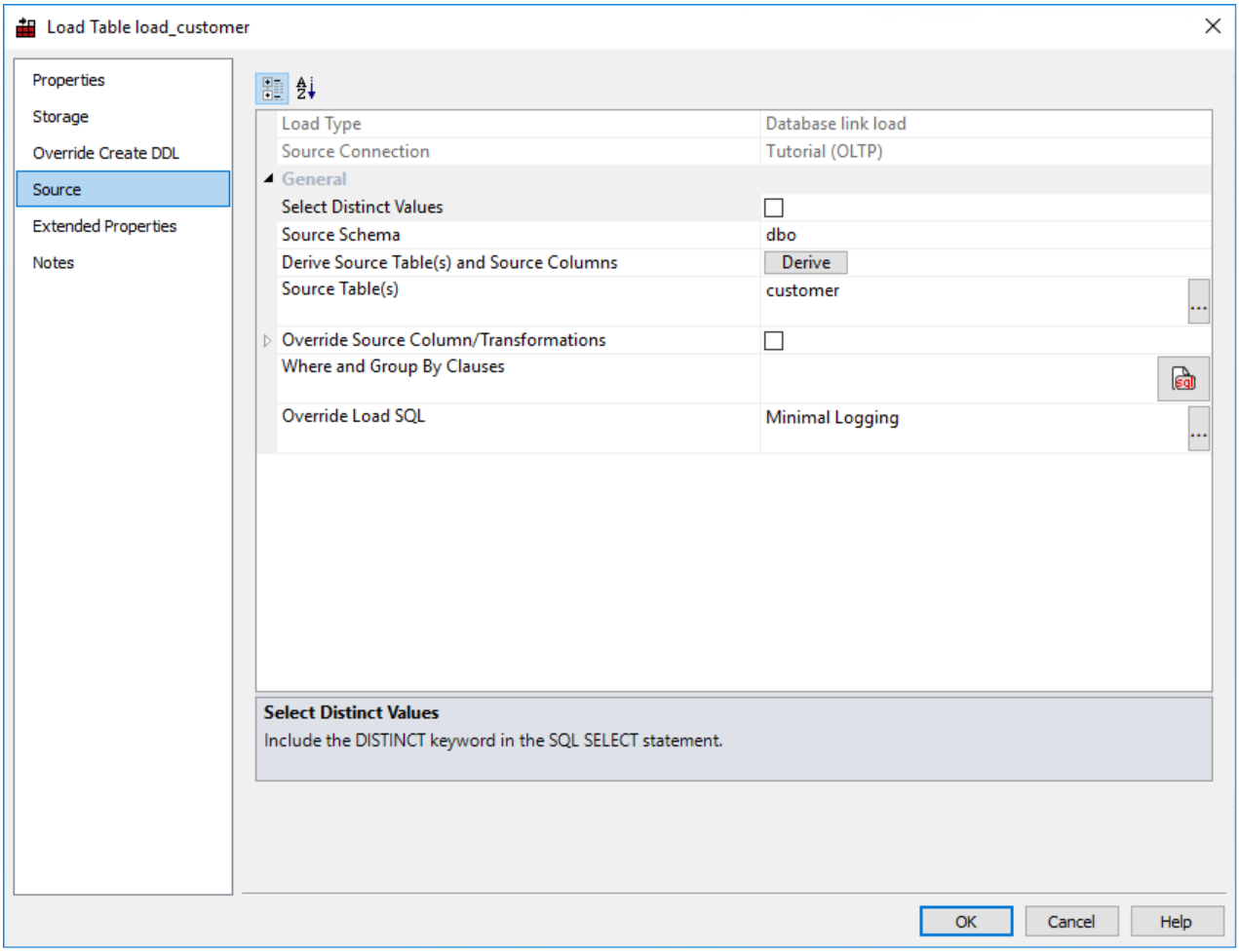
Fields	Description
Source Table(s)	Name of the table(s) where the data is sourced.
Override Source Column/Transformations	Ignore the source and transformation details of this table's columns and instead use the override details specified below. Refer to the section on **Load Table Transformations** below for details.
Allow Missing Source Columns	**Oracle Only** – Enables the load to proceed, even if one or more of the source columns does not exist. (refer to **handling missing source columns** for details).
Exit Status when Missing Source Columns	Load exit status when source columns are missing. Only enabled when **Allow Missing Source Columns** is set.
Where and Group By Clauses	Optional SQL SELECT WHERE-clause and/or GROUP BY-clause. Parameter names can be specified using $Pparameter_name$ (with leading $P and trailing $), which are replaced at run-time by the parameter's value. **Tip** This is where you can build a statement to handle change data. Parameter values can be in-line replaced and included in the 'Where' clause. Prefix the parameter name with a '$P' and add a trailing '$'. For example, if we have a parameter called SALES_LOCATION_CODE we could construct a statement, such as WHERE location_code = '$PSALES_LOCATION_CODE$' AND region_code = 'NY'. When this statement is executed, the value of the parameter will replace the parameter name. For example if the parameter was set to 'New York' then the statement would execute as: WHERE location_code = 'New York' AND region_code = 'NY'. Clicking the SQL button opens the **Where and Group By Clauses** editor which provides control settings to assist you in constructing the SQL statement. 
Override Load SQL	Optional SQL statement to load data into the table, which overrides all other properties. The specified SQL will be executed instead of generated SQL. For a linked database specify a complete INSERT statement. For an ODBC source, specify only the SELECT statement.
SQL Insert Hint	**SQL Server Only** - Optional hint to include in the SQL insert statement.
DB2 Load Logging	Controls the level of logging during the load (Minimal Logging or Recoverable Load).

© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
	The default is to reduce logging, which does NOT provide for full recovery.
DB2 Load Modifier	Optional Modifiers for the Load utility.
Source View Name	**Oracle only option**: Name of the database view to be created by WhereScape RED (if necessary) in the source system that will be used to select the source data. Refer to **Remote View Extract** for details.

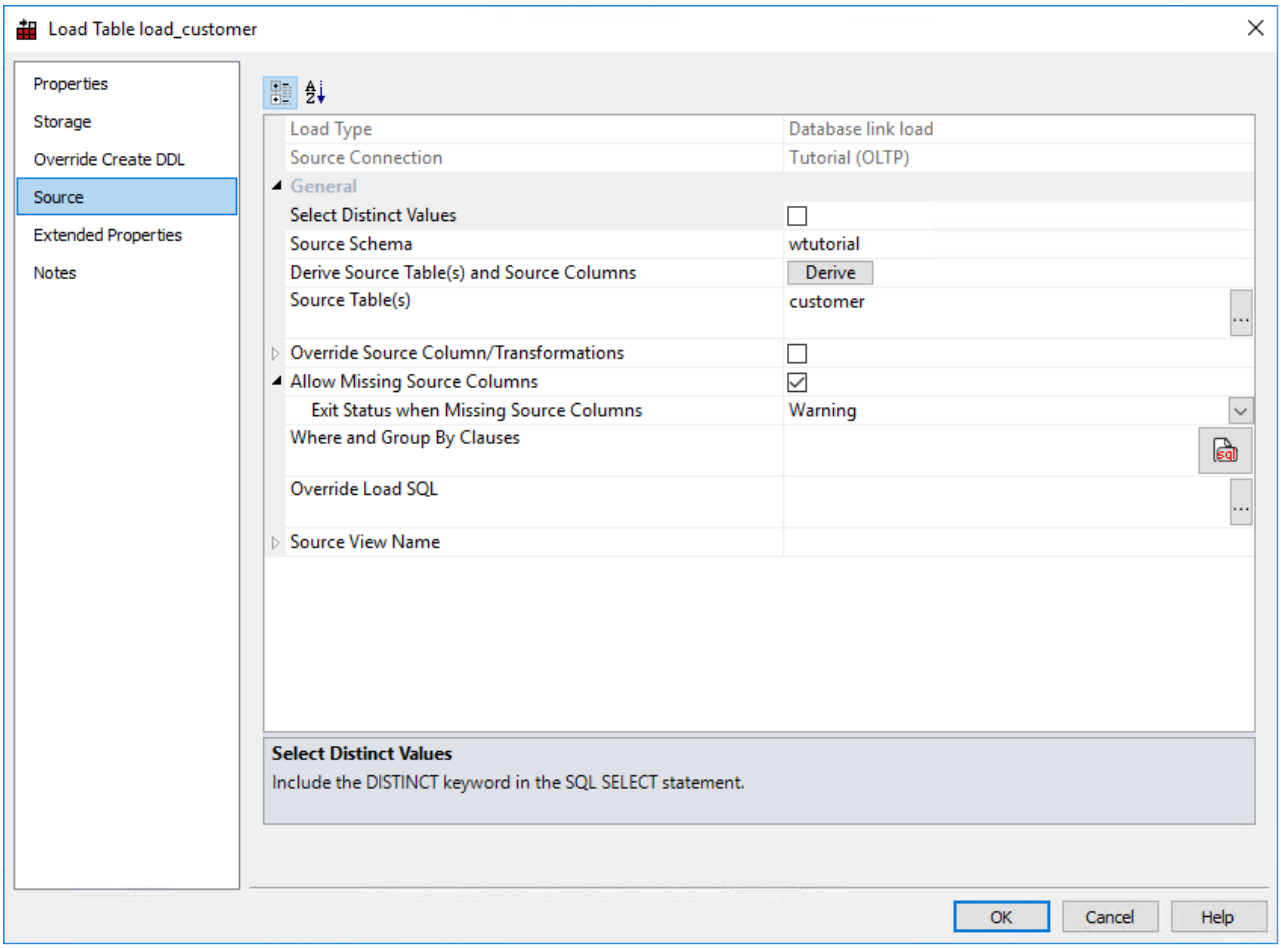
Load Table Source Mapping Screen

SQL Server Source screen:



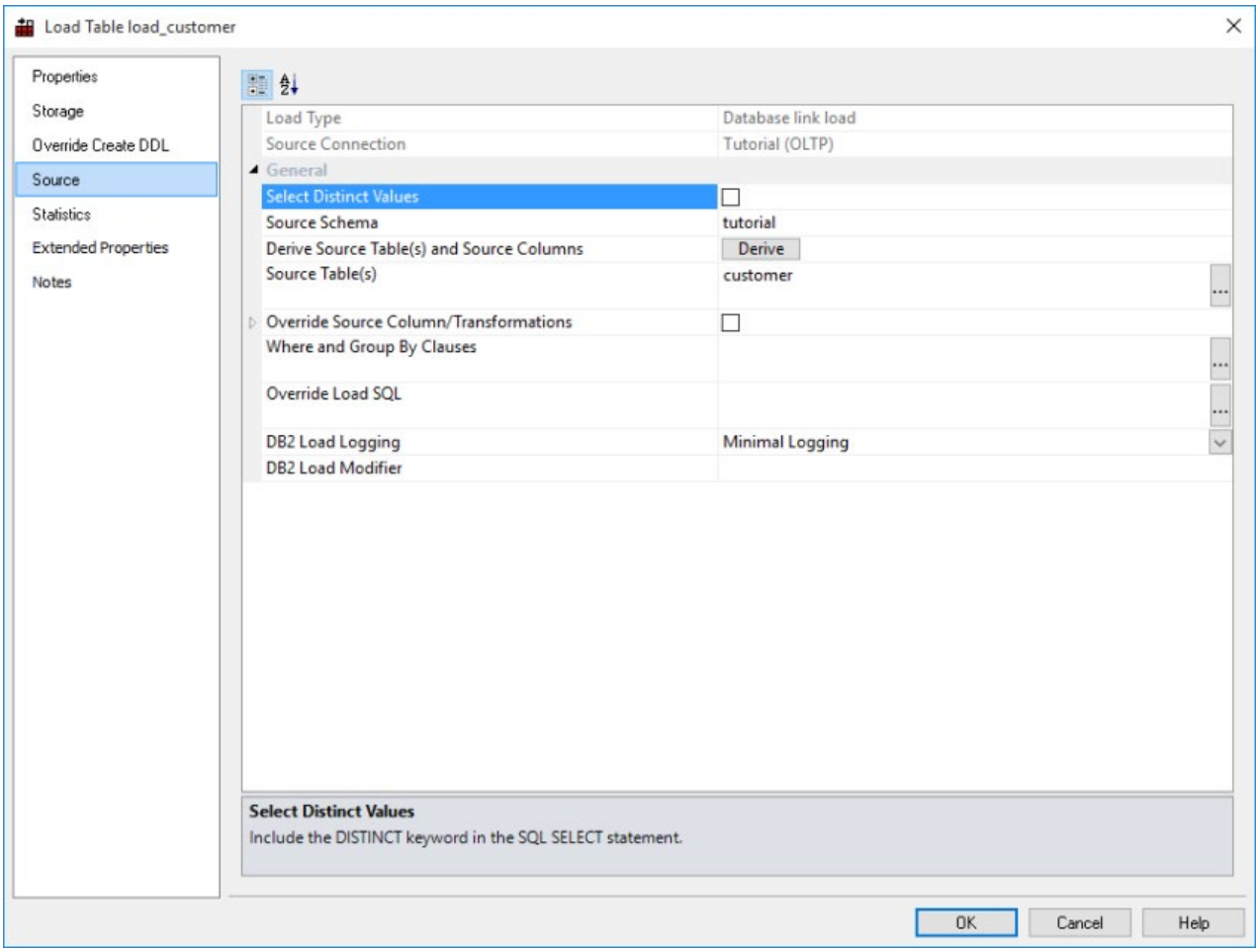
© Copyright, WhereScape Software Ltd All rights reserved.

Oracle Source screen:



© Copyright, WhereScape Software Ltd All rights reserved.

DB2 Source screen:



© Copyright, WhereScape Software Ltd All rights reserved.

Remote View Extract - Oracle Databases Only

Note

To use **Remote View Extracts**, both the **source** database and **target** data warehouse database must be **Oracle**.

Remote view extracts are used to force a remote Oracle source system database to perform complex extraction logic in the remote source system database and return the results to the data warehouse.

When specifying a join in an extract (load) over a database link, the Oracle cost based optimizer decides whether it is better to join the tables in the remote machine; or if the tables should be copied onto the local machine and the join performed locally. If the decision to copy is made for large tables, this can result in huge extract times. The simplest method of forcing the join to occur on the remote machine is to create a view on the remote machine from which the extract is done.

The view is re-created every time the load table is run. The load table then selects all columns from the remote view and inserts the results into the load table over the defined database link.

To create a remote view load table, users need to create a Database Link connection to load the remote view table from that connection. Refer to **Creating a Database Link** for details.

The steps for creating Database Link connection and then creating a remote view Load table are:

1. Create a connection object for the remote system or use an existing connection, populating the following fields:

Fields	Description
Connection Name	Name for the connection.
Connection Type	Set the connection type to **Database**.
ODBC Source	The ODBC source used to extract the source data.
Database ID	The Oracle SID or TNS Name.
Database Link Name	The name for the database link. This is a required field for establishing the **Database Link** connection.
Extract User ID	The user name that has access to the source database.
Extract User Password	The password for user that has access to the source database

2. Ensure the user name and password being used to connect to the remote system have the following privileges:
   - create view (granted as a specific privilege and not part of a role)
   - compile procedure (granted as a specific privilege and not part of a role)
   - execute on sys.dbms_sql
3. Right-click the Connection to the remote system and select **Create Database Link**. When the prompt database link has been created is displayed; click **OK**.
4. Right-click the Connection to the remote system and select **Create Remote View Procedure**. This creates the remote view procedure in the linked database.
5. When the prompt with the Procedure dss_view_create compiled is displayed, click **OK**.
6. Browse the Connection in the right pane.
7. Drag and drop a Source table or columns from the source tables to create the required Load table.
8. Create the Load table in the data warehouse database by selecting **Create (Recreate)** from the right-click context menu.
9. Right-click the Load table and select **Source**.
10. Enter a name for the remote view to be created into the **Source View Name** field. For example, use the name of the Load table.
11. Cut or copy any existing 'Where' clause from the **Where and Group By Clauses** text box and paste it into the **Source View WHERE-clause** text box. Alternatively, enter the 'Where' clause (including any table joins, if multiple Source tables are being extracted from) into the **Source View WHERE-clause** text box.

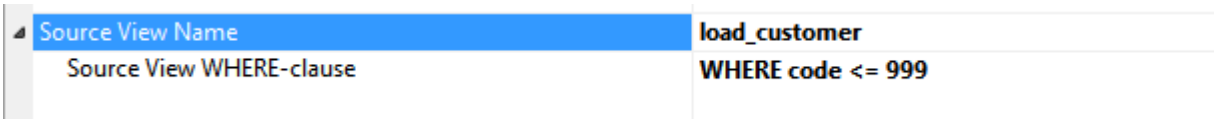
© Copyright, WhereScape Software Ltd All rights reserved.

Note:

Once the Load table has a remote view name, the 'Where' clause in the **Where and Group By Clauses** text box is ignored and the 'Where' clause in the **Source View WHERE-clause** box is used instead.

12. Click **OK**.
13. Load or schedule the Load table update.

Example remote view Load table source mapping properties:



Note:

During the Load operation, column transformations are ignored with Remote View Extract load tables. After the Load operation, column transformations can be used.
To create a derived column on a Remote View Extract load table, add the column leaving its source table blank and setting its source column to NULL. Then create an After Load transformation to populate the derived column.

ODBC Based Load

An ODBC based load provides an extensive option for acquiring data from many sources. It will be slower than most other forms of loads but may still be a viable option depending on the volume of data being loaded.

An ODBC based 'interactive load' when using the WhereScape RED tool will use the established ODBC connection to pull the data back to the local PC and then push the data to the data warehouse database via a sql ODBC statement.

A scheduler load will perform in the same way except that the data is loaded into the server that is running the scheduler and then pushed to the data warehouse database.
For Oracle, this push to the data warehouse can be performed via Oracle Direct Path loading, which can be considerably faster than the ODBC Sqll method.

For Oracle Direct Path loading to work all dates and times must be a character string of the form 'YYYY-MM-DD HH24:MI:SS'.

This is normally achieved via a 'During' load transformation such as
`TO_CHAR(source_date_column,'YYYY-MM-DD HH24:MI:SS')`

Note

Dates must be in the correct format for Oracle Direct Path loading to occur.

The obvious disadvantage in an ODBC based load is the two network transactions required and the overhead placed on the Scheduler server. The UNIX scheduler does not support ODBC based loads, therefore a Windows scheduler must be available to handle ODBC based loads.

Note

A Windows scheduler must be available to handle ODBC based loads.

The **Properties** screens for an ODBC based load are the same as those of a database link load. Refer to the previous sections for details.

© Copyright, WhereScape Software Ltd All rights reserved.

SSIS Loader

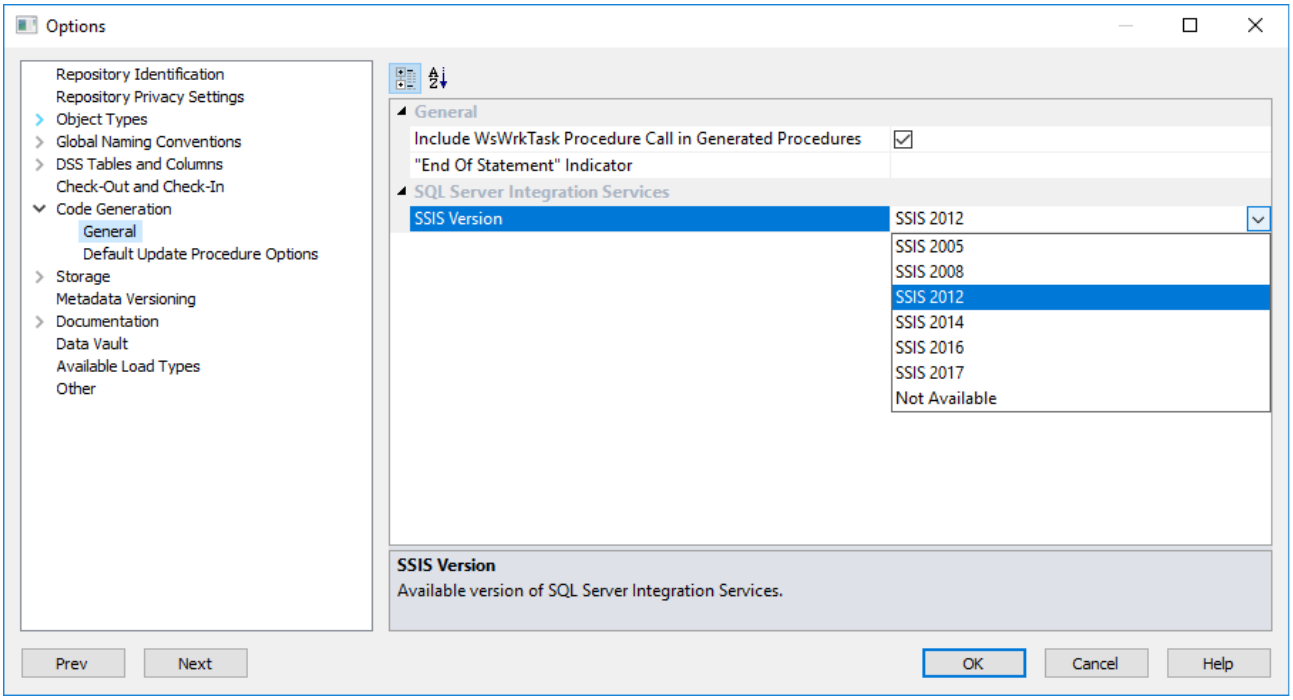
Microsoft SQL Server Integration Services (SSIS) is an Extract Transform and Load (ETL) utility that is supplied and licensed as part of Microsoft SQL Server 2005+. SSIS is built to extract data from data sources using extraction technologies, such as OLEDB, transform the data using its own .NET based language and then load it into data destination. SSIS is primarily used for loading data from various sources into SQL Server, but it can be used to extract from most databases and load into any other database.

WhereScape RED manages the extraction and load of data into the data warehouse as part of data warehouse processing. WhereScape RED attempts to utilize the optimum loading technology for each supported database platform. Optimal loading performance from a database source into SQL Server is achieved using SSIS. WhereScape RED provides the ability to leverage the extract and load performance available from SSIS as part of the WhereScape RED processing. WhereScape RED does this by generating and executing an SSIS package dynamically at run time. Any errors or messages resulting from execution are passed back into the WhereScape RED workflow metadata to drive subsequent workflow actions. In the event of an error during execution of the SSIS package, the package will be persisted to disk to assist the developer with problem resolution.

An **Integration Services Load** can have a number of source databases:

- SQL Server
- Oracle
- DB2 for Linux, UNIX and Windows (the DB2 RED can build a warehouse in)
- Teradata
- Sybase
- MySQL
- DB2/400 (the DB2 that runs on AS/400 machines - quite different to above)
- MS Access
- Excel

To use SSIS to load data, select the relevant version of SSIS in **Tools > Options > Code Generation > General**.

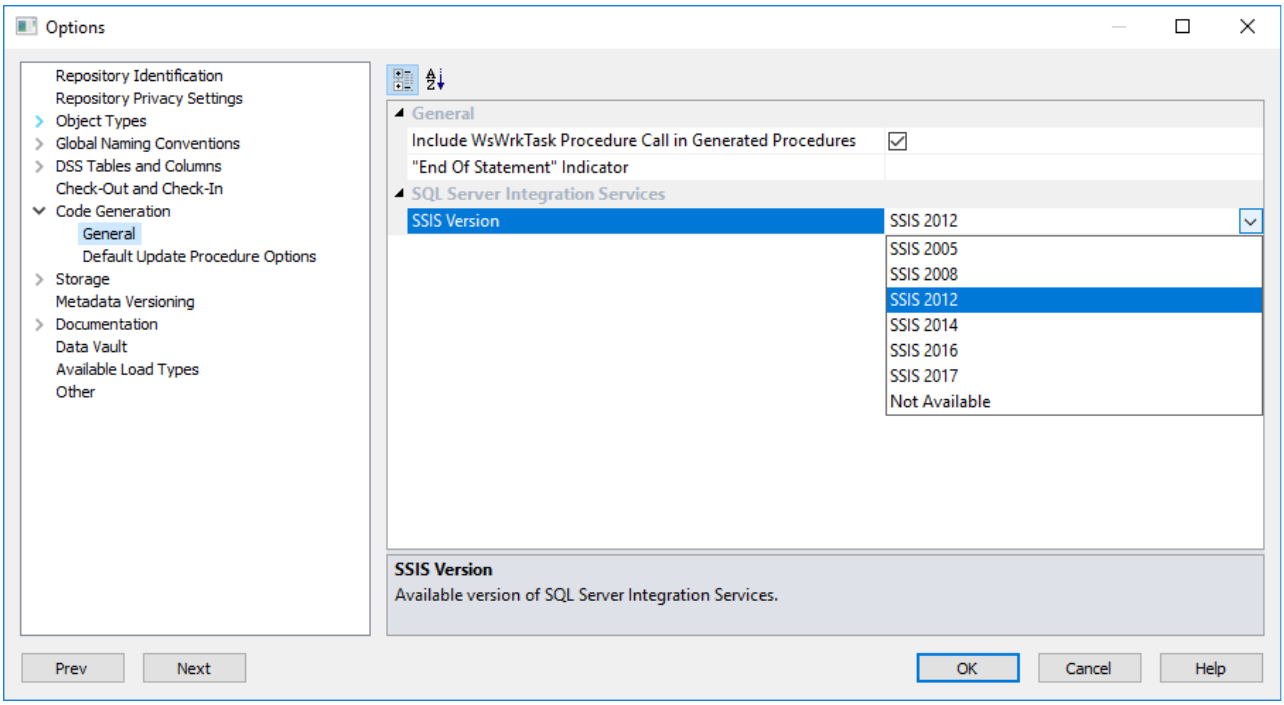


© Copyright, WhereScape Software Ltd All rights reserved.

Loading data into RED load tables using SSIS

To use SSIS loading, ensure that SSIS loads are enabled and that the SSIS version is set correctly in **Tools > Options > Code Generation > General**.

Refer to **Loading Data from a Flat File using SSIS** for details, if you are loading data from a Flat File source.



RED can extract and load data using SSIS from database tables or flat files from a Windows connection. As with any load into RED, a connection to the source data needs to be created to provide extraction details.

The **SSIS Connection String** is a valid SSIS connection string that can be used to connect to the data source or destination.

Loading Data via SSIS from a database

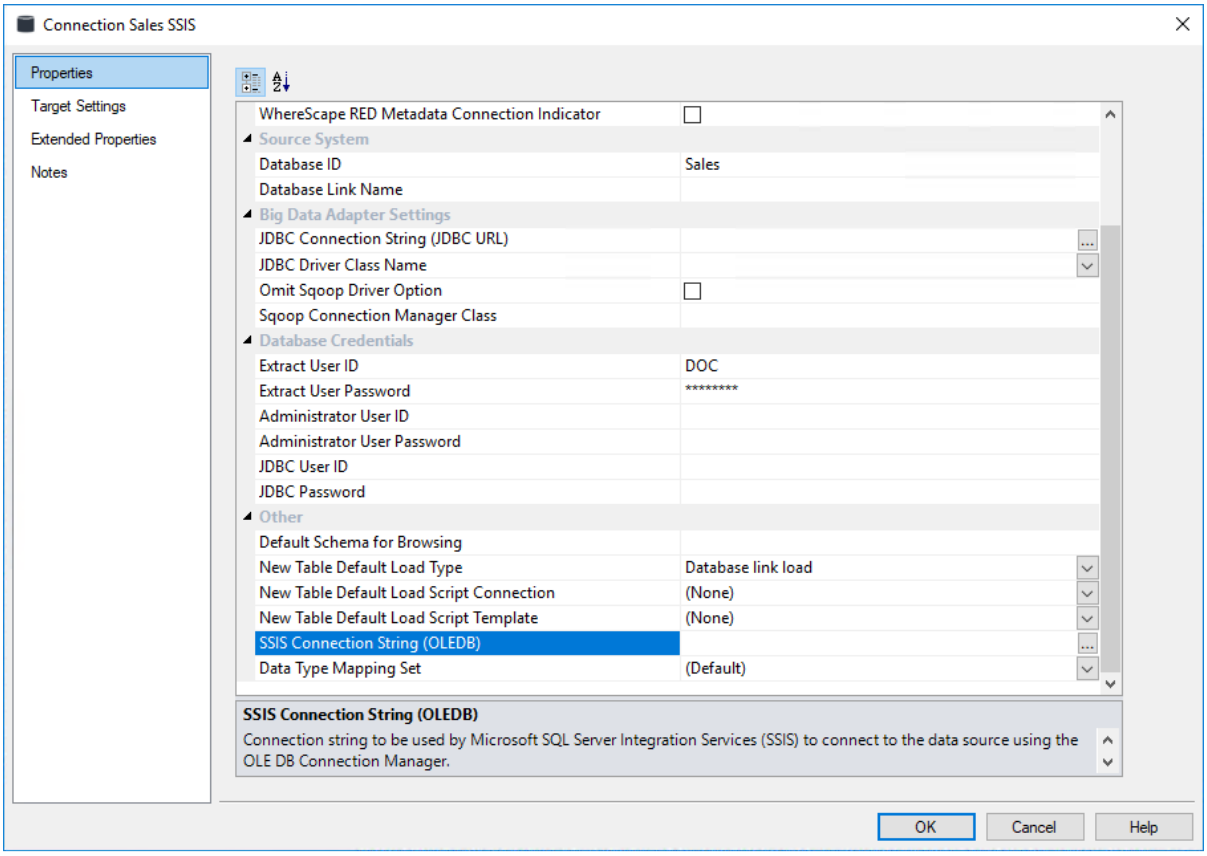
If the connection is a database load ,then there is additional connection information that should be supplied to use SSIS as a loading option.

This additional information needs to be supplied on both the source connection and the data warehouse connection.

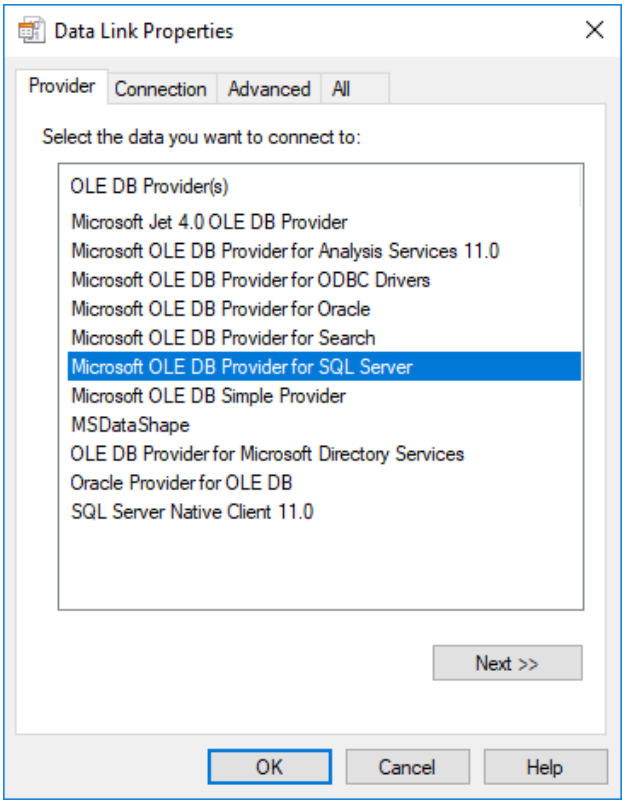
© Copyright, WhereScape Software Ltd All rights reserved.

Creating the SSIS Connection String

1. Click the **ellipsis** button in the right side of the SSIS Connection String field of the relevant connection:



2. On the **Provider** tab, select the relevant **OLE DB Provider** and click **Next**.

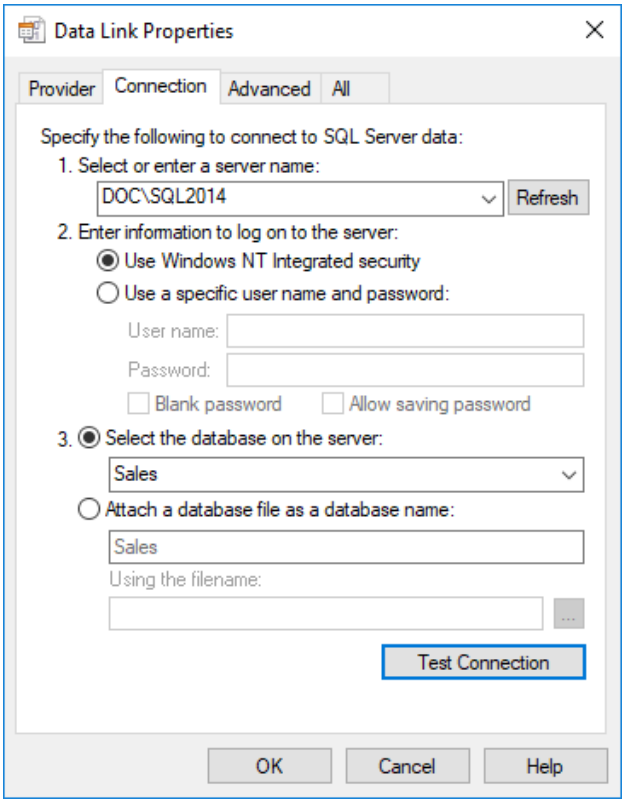


3. On the **Connection** tab, select the **server name**, enter the **information to log on to the server** and select the **database** on the server. Click **Test Connection**.

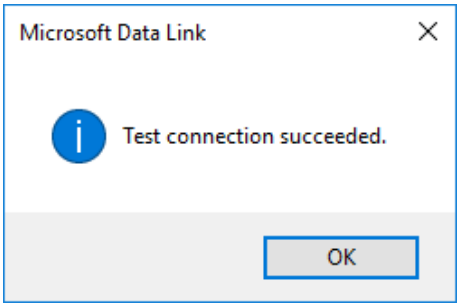
© Copyright, WhereScape Software Ltd All rights reserved.

Notes:

- When using a specific **user name** and **password** to connect to the server instead of using Windows integrated security, the **Allow saving password** check-box must be selected.
- It is also recommended that the password on the SSIS connection string field that is displayed in the connection properties is replaced with the **$PASSWORD$** token that is substituted at runtime.

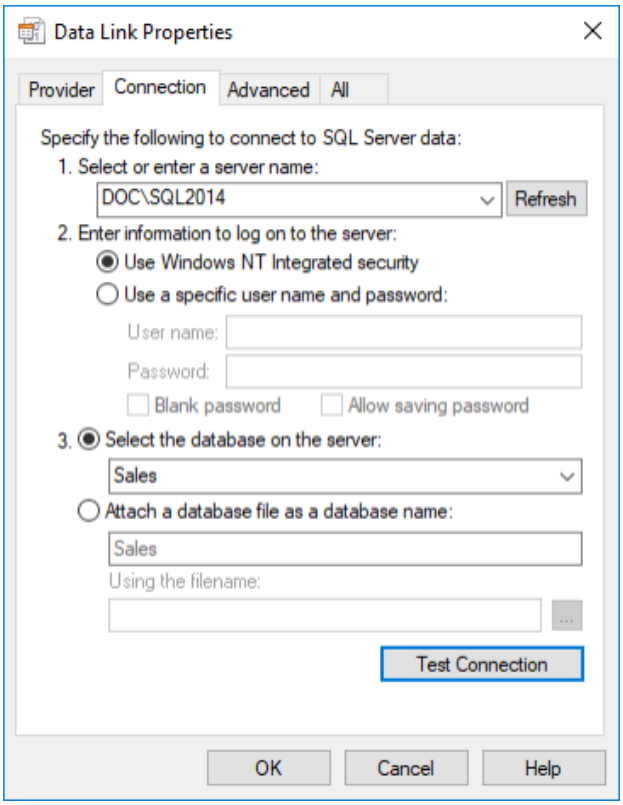


4. Click **OK**.

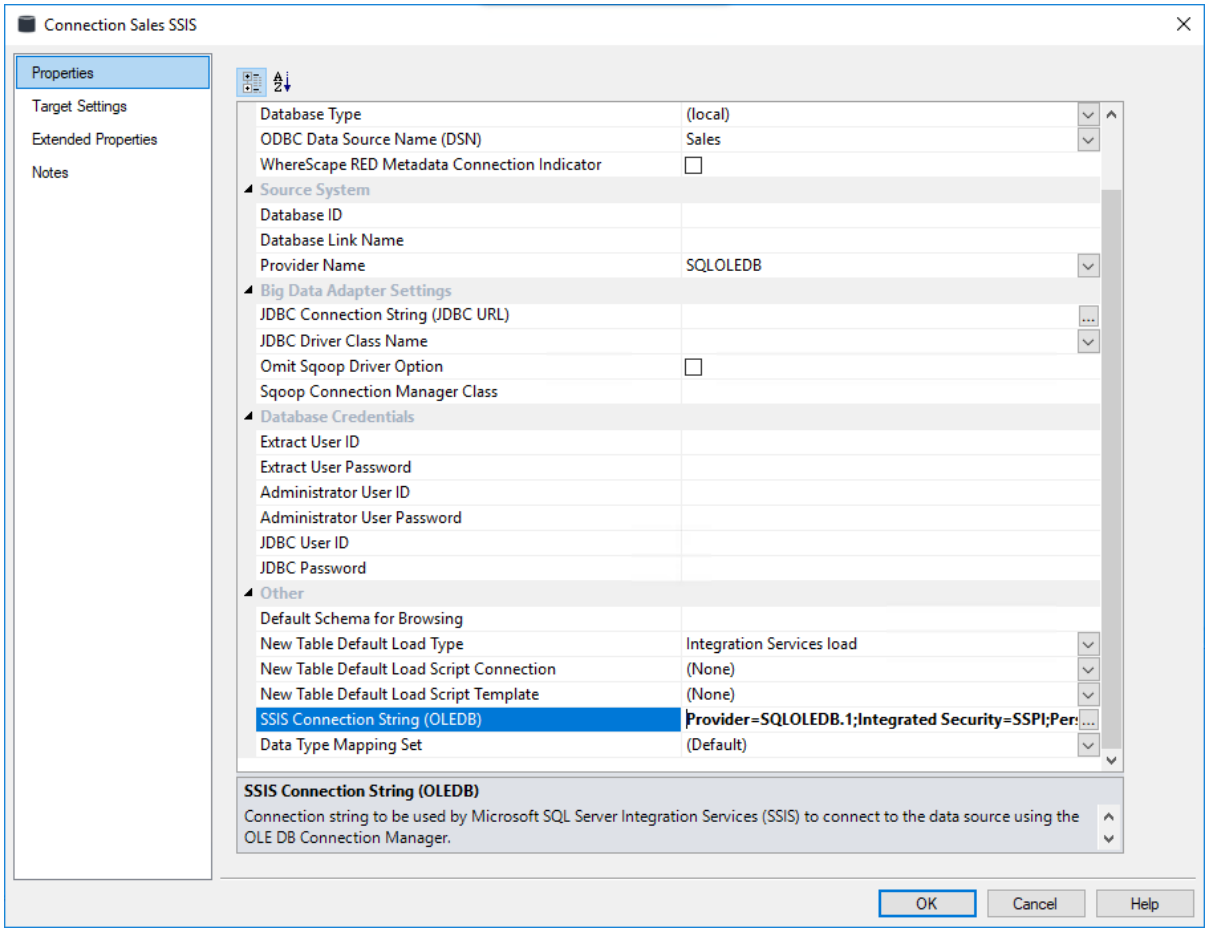


© Copyright, WhereScape Software Ltd All rights reserved.

5. Click **OK** in the **Data Link Properties** window to save the SSIS connection string settings.



6. The SSIS connection string is displayed.

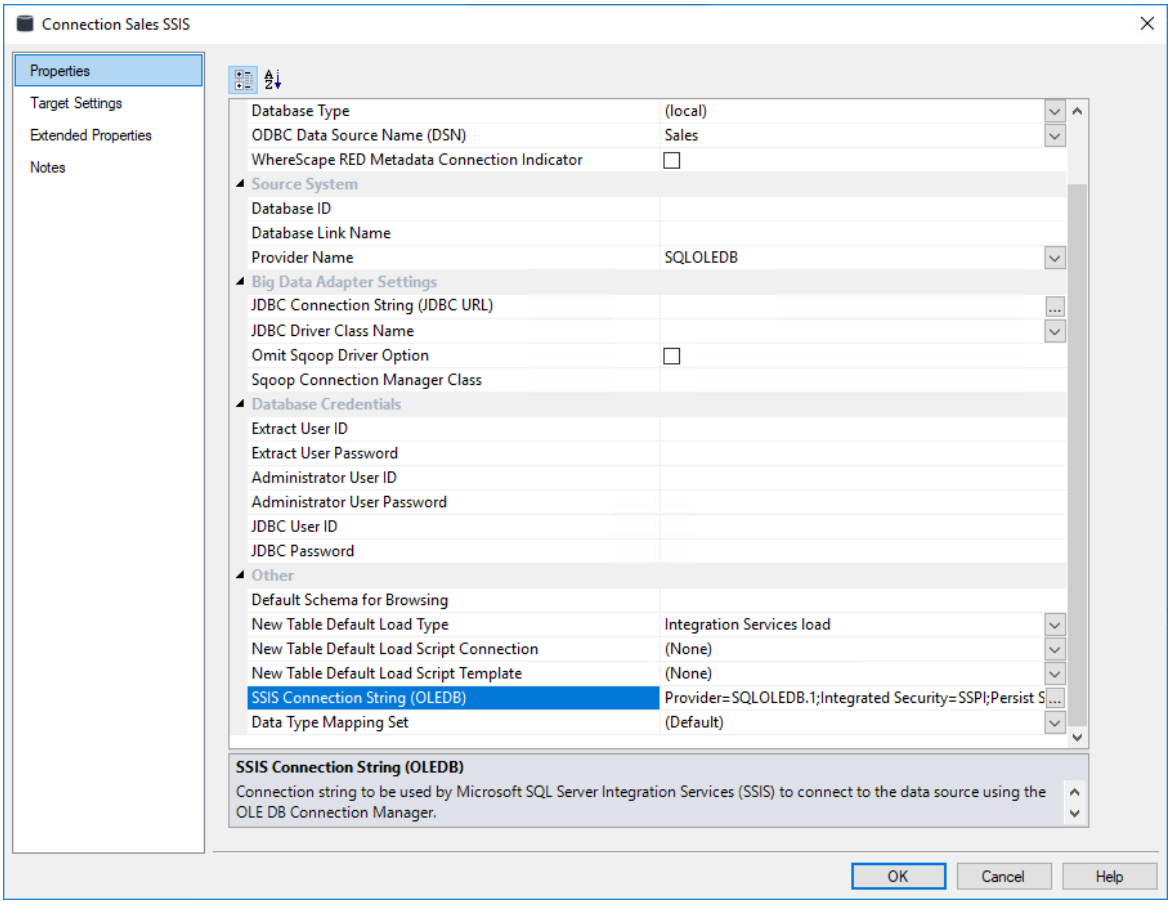


7. Click **OK** to save the connection.
   - Right-click **Sales SSIS** and select **Browse Connection**.

© Copyright, WhereScape Software Ltd All rights reserved.

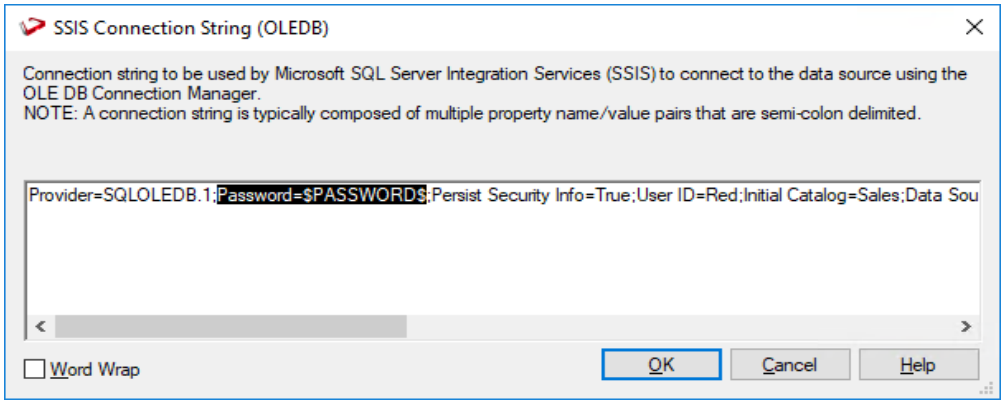
- Accept the defaults and click **OK**.

8. In SSIS terms, you have now defined your **Source** in **SSIS Connection Manager**. Using the same process, you need to add the SSIS Connection String to the data warehouse connection so that you can specify your **Destination** connection:
   - Double-click the **DataWarehouse** connection in the object explorer to open up the Properties screen.
   - Follow the process above to create the SSIS Connection String.
   - Click **OK** to save your connection.



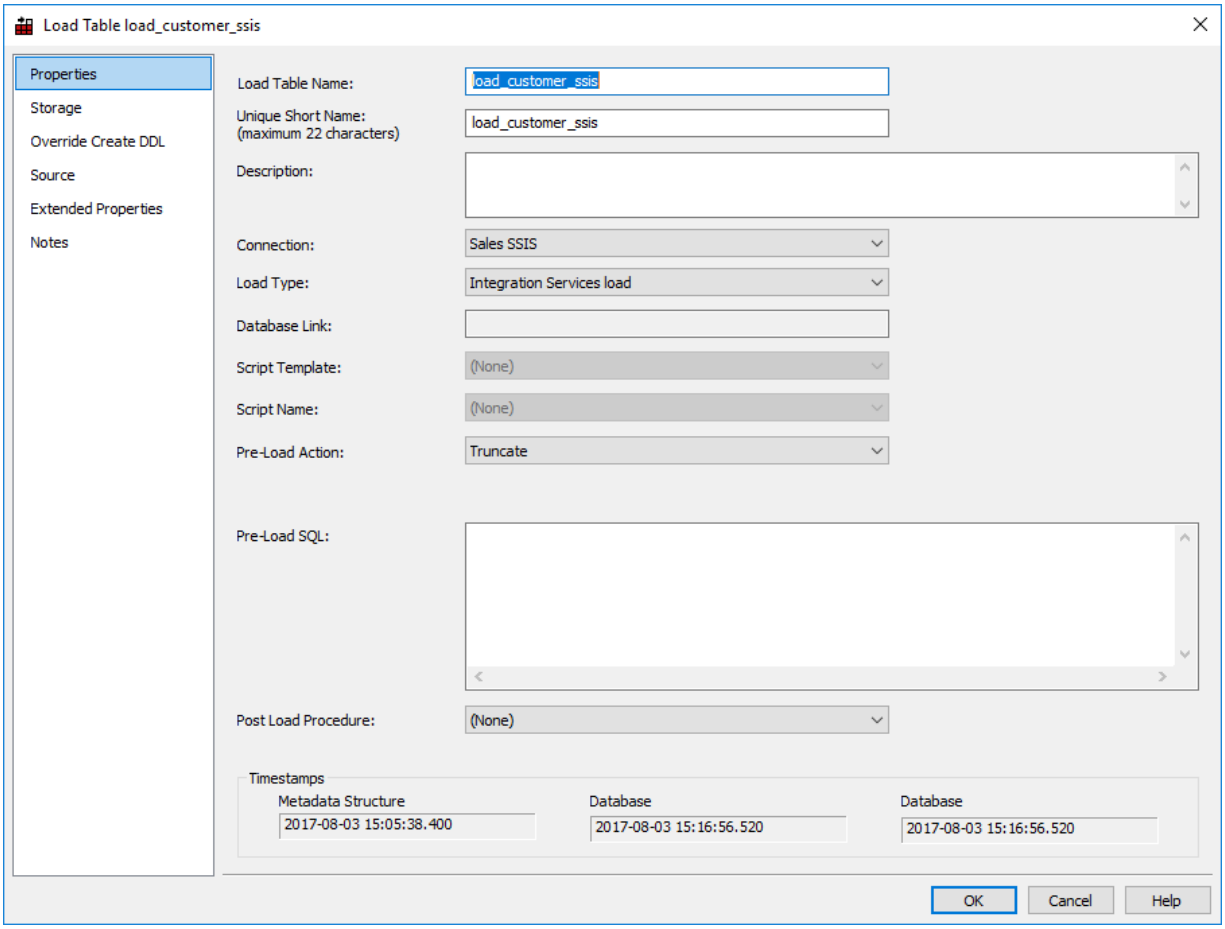
Notes:

- If the connection string is already set, then the ellipsis button will open up an editor dialog. Edit the connection string and click **OK**..
- For connections that require a username and password, the connection string can also be edited to replace the password with the **$PASSWORD$** token that is substituted at runtime. If the $PASSWORD$ token is used, RED uses the contents of the masked "Extract User Password" property when making the connection. E.g. "Provider=SQLOLEDB.1;Password=**$PASSWORD$**;..."



© Copyright, WhereScape Software Ltd All rights reserved.

9. Once the connection is defined, a Load table must be created to hold data loaded into the data warehouse by dragging a source table or a flat file in the middle pane to create a Load table. (Refer to **Loading data** and **Flat File Load** for details.)
   The Load type can be set to Integration Services load from the Load table properties window. This creates and executes a SSIS package at run time, to load data into the data warehouse Load table.



The configuration options available on an SSIS load are available in the **Source** tab of the Load table's Properties. These options are detailed in the next topic SSIS Load Table Source Screen.

SSIS Load Table Source Screen

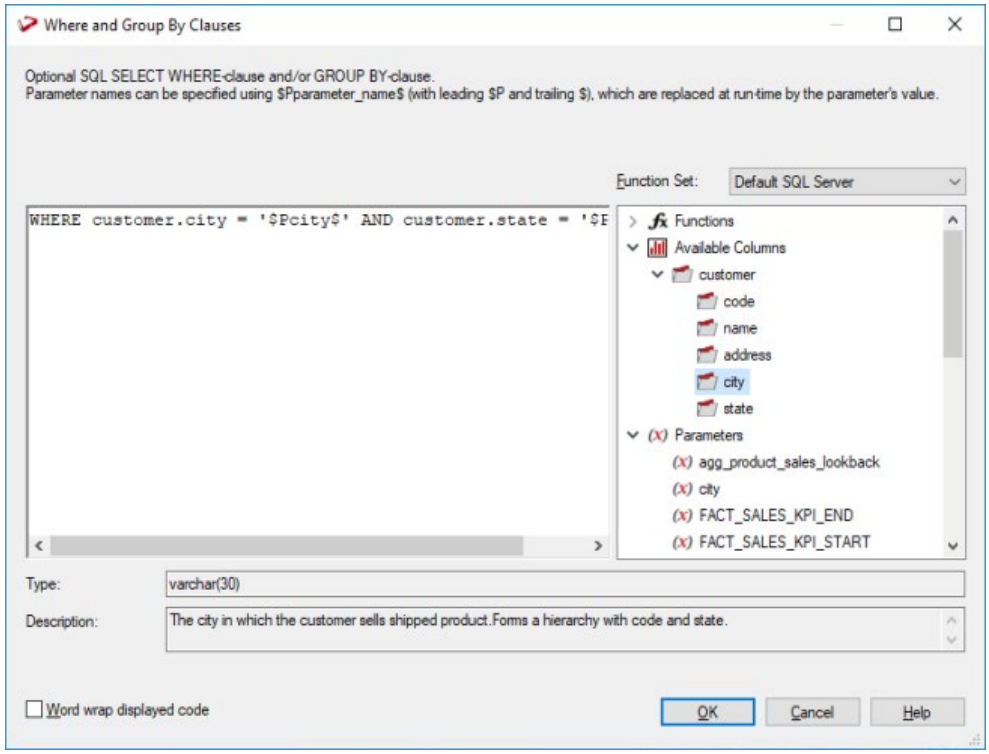
The configuration options available on an SSIS load are available on the **Source** tab of the Load table's Properties. These options are:

Fields	Description
Load Type	Method of loading data into the table. The available options are dependent on the **Source Connection**. Defaults to the 'Default Load Type' of the **Source Connection**. Can be specified via the Properties screen.
Source Connection	The connection that identifies the source database. Can be specified via the Properties screen.

General

Fields	Description
Select Distinct Values	Include the DISTINCT keyword in the SQL SELECT statement.
Source Schema	Schema within the source database where the source table resides.
Derive Source	Derive the **Source Table(s)** and **Source Column(s)** properties (of this dialog) from

© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
Tables(s) and Source Columns	the source details of this table's columns. **Note** The existing property values will be overwritten.
Source Table(s)	Name of the table or tables that the data is sourced from.
Override Source Column/Transformations	Ignore the source and transformation details of this table's columns and instead use the override details specified below. Refer to **Load Table Transformations** for details.
Exit Status when Missing Source Columns	Load exit status when source columns are missing. Only enabled when **Allow Missing Source Columns** is set to True.
Where and Group By Clauses	Optional SQL SELECT WHERE-clause and/or GROUP BY-clause. Parameter names can be specified using $Pparameter_name$ (with leading $P and trailing $), which are replaced at run-time by the parameter's value. **Tip** This is where you can build a statement to handle change data. Parameter values can be in-line replaced and included in the 'Where' clause. Prefix the parameter name with a '$P' and add a trailing '$'. For example, if we have a parameter called SALES_LOCATION_CODE we could construct a statement, such as WHERE location_code = '$PSALES_LOCATION_CODE$' AND region_code = 'NY'. When this statement is executed, the value of the parameter will replace the parameter name. For example if the parameter was set to 'New York' then the statement would execute as: WHERE location_code = 'New York' AND region_code = 'NY'. Clicking the SQL button opens the **Where and Group By Clauses** editor which provides control settings to assist you in constructing the SQL statement. 
Override Load SQL	Optional SQL statement to load data into the table, which overrides all other properties. The specified SQL is executed, instead of generated SQL. For a linked

© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
	database, specify a complete INSERT statement. For an ODBC source, specify only the SELECT statement.
Allow Missing Source Columns	**Oracle only**: Allows the load to happen when one or more of the source columns do not exist. (Refer to **handling missing source columns** for details).

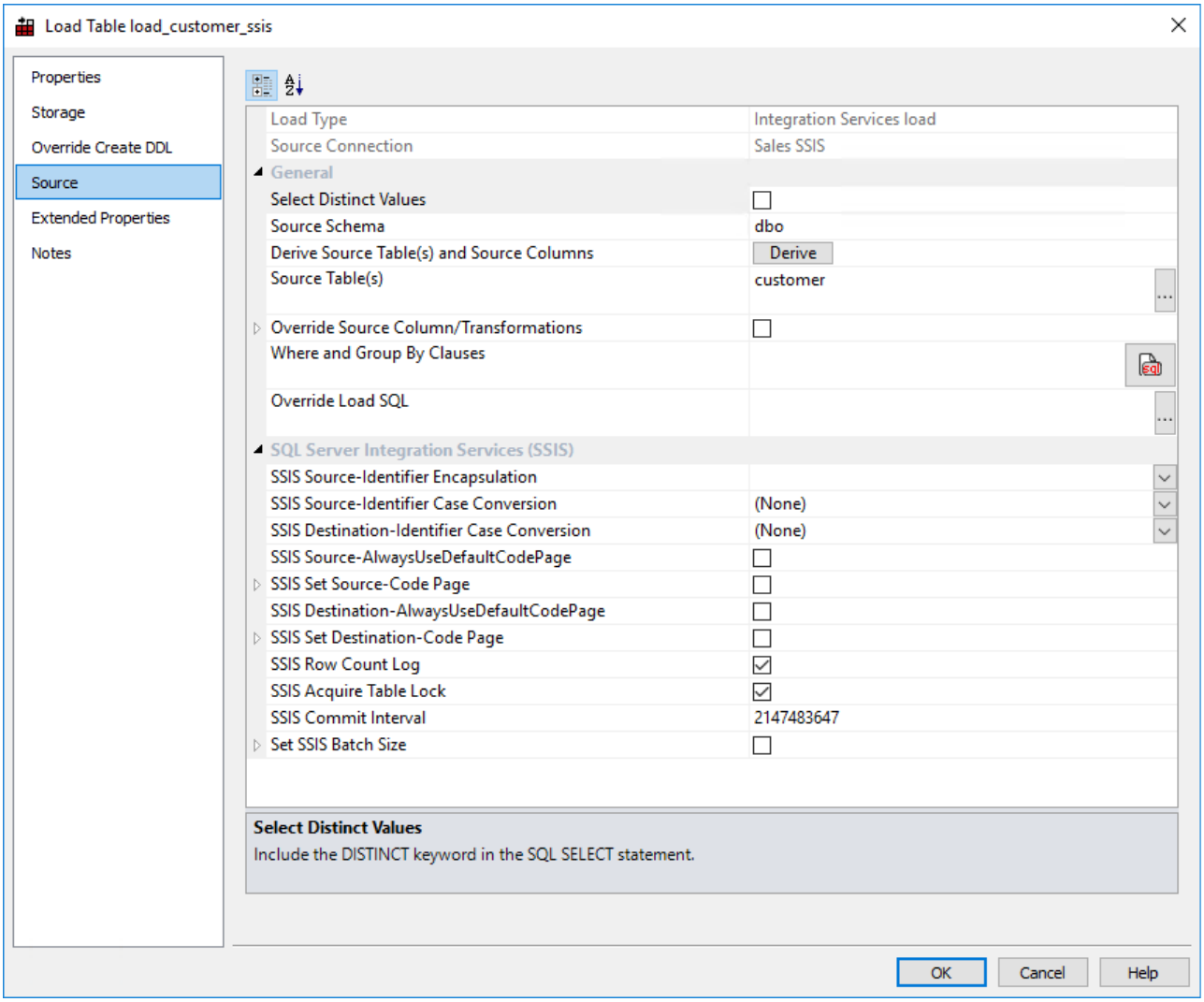
SQL Server Integration Services (SSIS)

Fields	Description
SSIS Source-Identifier Encapsulation	Characters that are used to enclose source column names. Options are (None), "", [], ', ``
SSIS Source-Identifier Case Conversion	Case-sensitivity conversion applied to Source Object Identifiers (such as table, view, and column names) in RED-generated SSIS packages. If no conversion is applied, then the exact case of the identifier defined in the RED metadata is used in SSIS.
SSIS Destination-Identifier Case Conversion	Case-sensitivity conversion applied to Destination Object Identifiers (such as table, view, and column names) in RED-generated SSIS packages. If no conversion is applied, then the exact case of the identifier defined in the RED metadata is used in SSIS.
SSIS Source-AlwaysUseDefaultCodePage	Forces the use of the DefaultCodePage property value when describing character data.
SSIS Destination AlwaysUseDefaultCodePage	**SQL Server**, **Oracle** and **Greenplum** - Forces the use of the DefaultCodePage property value when describing character data.
SSIS Set Destination-Code Page	**SQL Server**, **Oracle** and **Greenplum** - Enables the SSIS Destination Code Page property.
SSIS Row Count Log	During an SSIS Load, include Row Count logging.
SQL Server only options	
SSIS Acquire Table Lock	Specifies whether the destination table is locked during the load. This enables the SQL Server to reduce lock escalation overheads during loading and will promote minimal logging in SQL 2008+ when a Bulk Logged recovery model is used for the database. The default value of this property is true.
SQL SSIS Commit Interval	Specifies the batch size that the OLE DB destination tries to commit during fast load operations. The default value of 2147483647 indicates that all data is committed in a single batch after all rows have been processed. If you provide a value for this property, the destination commits rows in batches that are the smaller of (a) the Maximum insert commit size, or (b) the remaining rows in the buffer that is currently being processed.
SQL Set SSIS Batch Size	Specifies the number of rows in a batch. The default value of this property is empty, which indicates that no value has been assigned.
PDW only options	
SSIS Load Mode	Select the SSIS loading mode from the list. The available options are Append, Fastappend, Reload and Upsert. When the Upsert option is selected, the user must specify a key to use for the Upsert. The Upsert Key should be a business key for the loaded data.

© Copyright, WhereScape Software Ltd All rights reserved.

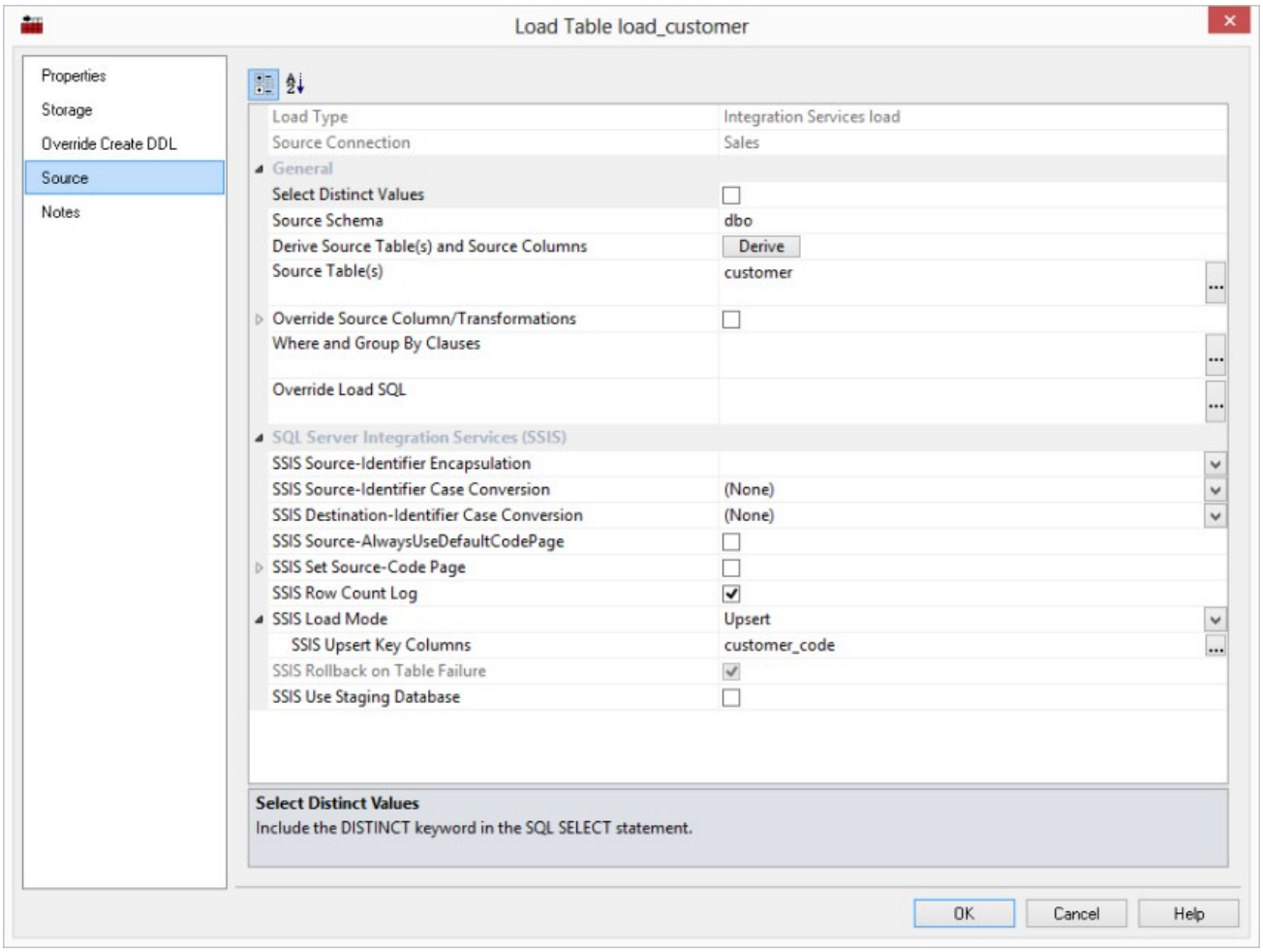
Fields	Description
	Notes • To use the Upsert feature, please ensure the pre-load action in the main properties screen is disabled. • PDW does not allow use of staging DB or Rollback if fastappend is the load mode chosen. • Fastappend cannot be used if table has the storage setting of replicated.
SSIS Upsert Key Columns	Columns selected to make up the Upsert Key. The Upsert Key should be a business key for the loaded data. This can be typed in as comma-separated list of column names or alternatively, users can use the column selection dialog by clicking the ellipsis button to the right of the SSIS Upsert Key Columns field, to select the desired columns.
SSIS Rollback on Table Failure	In the event of a SSIS failure, a rollback is performed.
SSIS Use Staging Database	If this option is set and a staging database is defined on the target connection, then the staging database is used instead.

SQL Server SSIS load Source screen example:



© Copyright, WhereScape Software Ltd All rights reserved.

PDW SSIS load Source screen example:



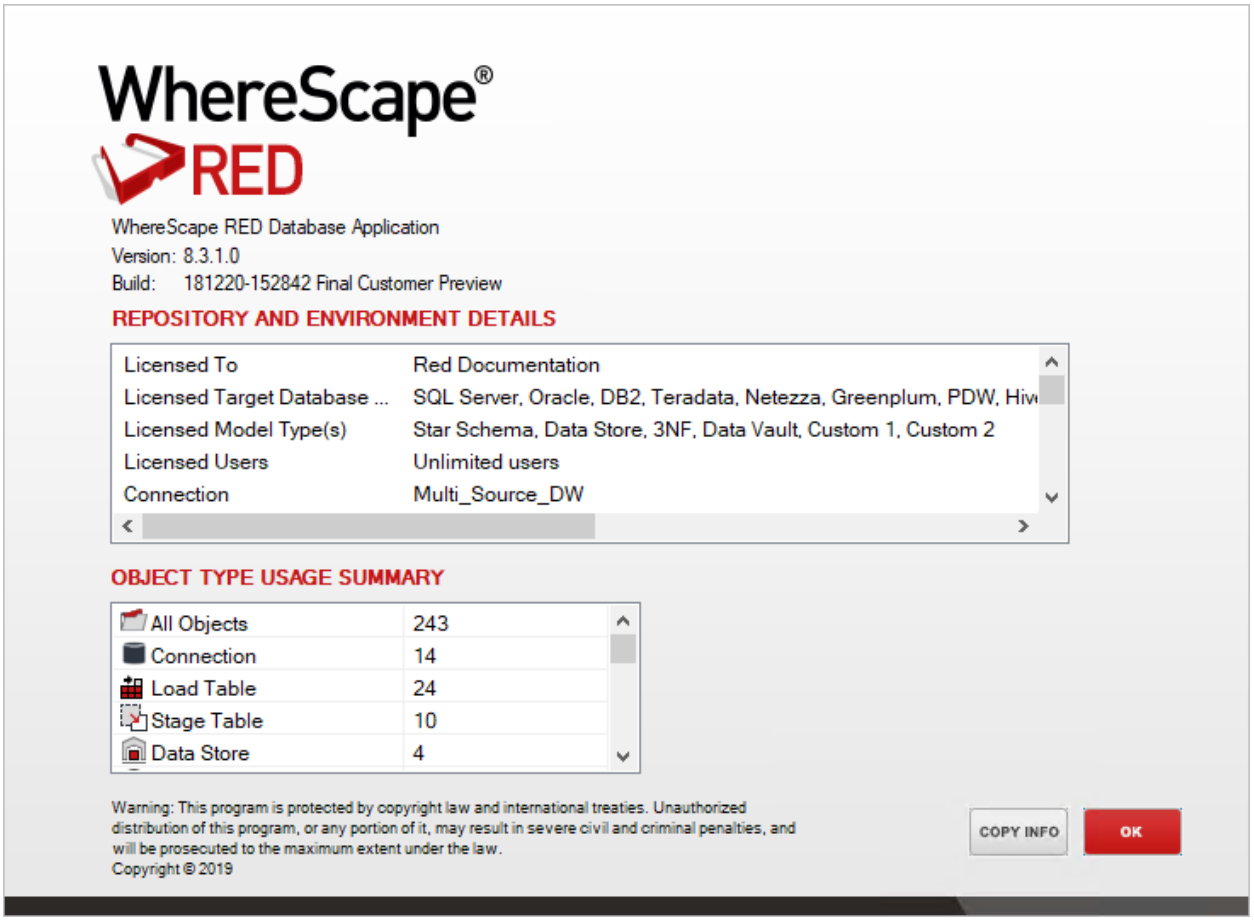
© Copyright, WhereScape Software Ltd All rights reserved.

Troubleshooting

If an SSIS load fails due to incorrect connection strings, then the SSIS package will fail to create at run time. The connection string information in both the source connection and the data warehouse connection (the destination) must be checked to ensure that they are correct.

If the SSIS load fails due to incorrect extraction SQL, due to mismatched destination Load table structure or due to constraint violation, then the SSIS package is created in the file system for the developer to check. The SSIS package file is created in the WhereScape RED Work Directory; shown in the window opened from **Help > About WhereScape RED** (refer to the screen below).

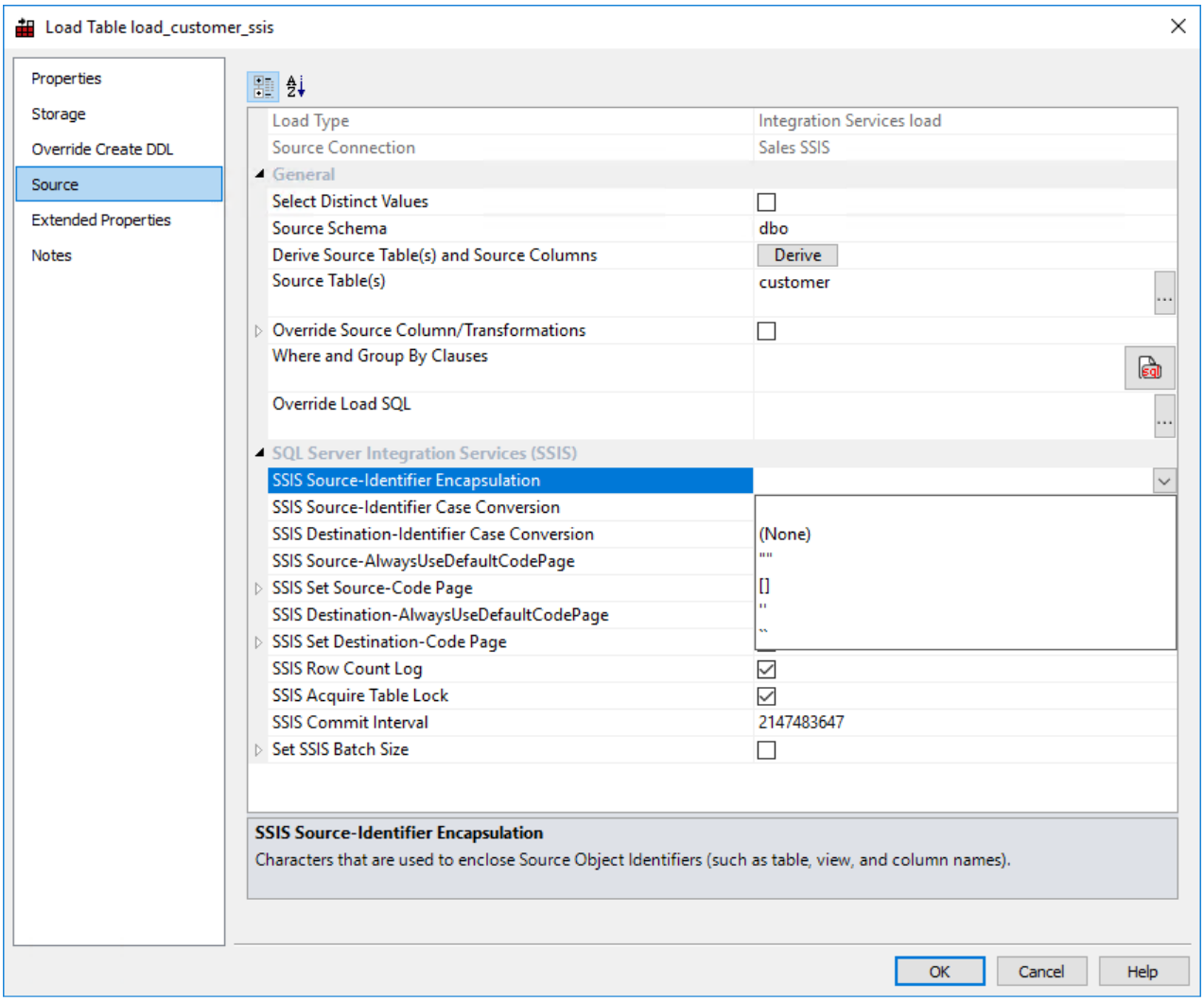
This SSIS package will have a DTSX file extension and can be opened using the appropriate version of Microsoft's BI Development Studio add-in to Visual Studio.



The **Integration Services load** in RED does not currently automatically handle spaces in source field names. You need to put square brackets([ ]) around the source column in the column Properties of the Load table column that contains spaces in Excel. This prevents the SSIS (runtime) from crashing and returning an error message. The SSIS package file is created in the WhereScape RED Work Directory; shown in the above window, opened from **Help > About WhereScape RED**.

The **Source** tab for a Load_table has a drop-down list that provides the options for **SSIS Source-Identifier Encapsulation**.
We use encapsulation to ensure that column aliases are not changed during an SSIS load:

© Copyright, WhereScape Software Ltd All rights reserved.



Native ODBC Based Load

A Native ODBC based load is similar to an ODBC load, except it provides a faster method to load data from non-linked databases.

A Native ODBC based **interactive load** when using the WhereScape RED tool will use the established ODBC connection to pull the data back to a delimited file on the local PC and then push the data to the data warehouse database via a Bulk Insert in SQL Server, SQL*LOADER in Oracle or LOAD statement in DB2 and Fastload session in Teradata.

A scheduler load will perform in the same way, except that the data is loaded into the server that is running the scheduler and then pushed to the data warehouse database.

All loaders require dates in the default format of the target data warehouse database. In Teradata, all dates and times must be a character string of the form 'YYYY-MM-DD HH24:MI:SS' for Fastload loading to work. Both of these are normally achieved via a **During** load transformation, using the correct casting function for the source database.

The **Properties** and **Storage** screens for a Native ODBC based load are the same as those of a database link load. Refer to the previous sections for details. Details of the source mapping screen follow.

Notes:

- The Native ODBC load in an ODBC connection is not enabled by default for newly created Teradata repositories. Refer to **Tools > Options > Available Load Types** for details.
- If you are doing Native Loads using UNIX and LINUX, see section Native Loads using UNIX and LINUX for more details about this type of load.

© Copyright, WhereScape Software Ltd All rights reserved.

Native Loads in Oracle

Some versions of Oracle do not return the ORACLE_SID correctly when an ODBC call requests it. To ensure that you don't have any issues because of this, please ensure that the following steps are complied:

If you are doing interactive Native Loads in RED into Oracle:

Ensure that the **Database ID (SID)** field in the Data Warehouse connection Properties screen is set to the actual ORACLE_SID for your database.

This value is used in the sqlldr connection during the native load IF the ORACLE_SID is not returned correctly by the ODBC driver.

If you are using the Windows scheduler for Native Loads into Oracle:

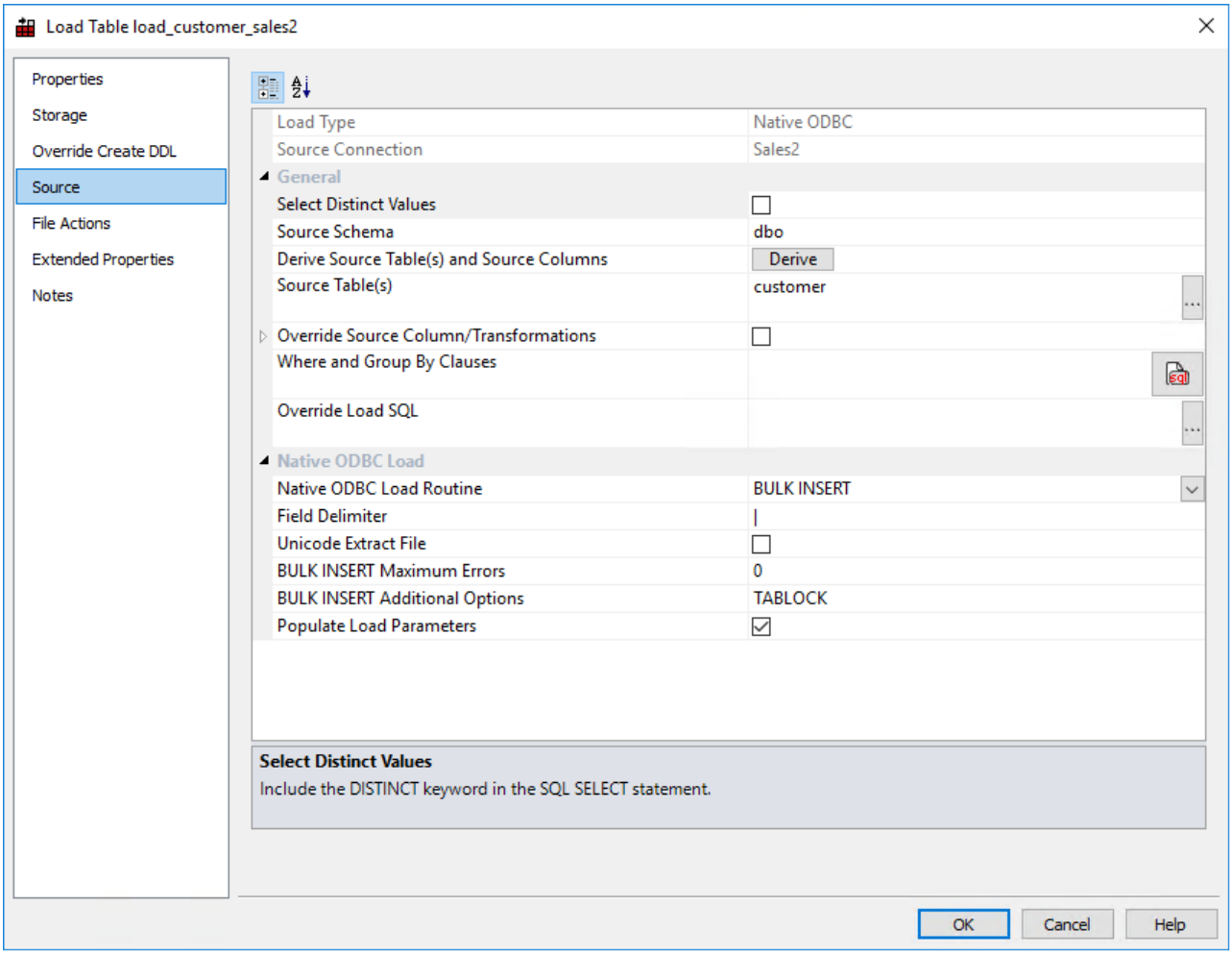
1. Change the scheduler to be a direct path load (via maintain scheduler in the WhereScape Administrator).
2. Ensure that the TNS Service is set to the actual ORACLE_SID for your database.
3. Restart the scheduler.

This value is used in the ORACLE sqlldr connection IF the SID is not returned by the ODBC driver and only IF the scheduler is set to do direct path loads.

Native ODBC Based Load - Source Screen

The fields on the **Source** tab for Native ODBC loads are described below. **Oracle** and **DB2** Source screen examples can also be found at the bottom of this topic.

SQL Server Source Screen:



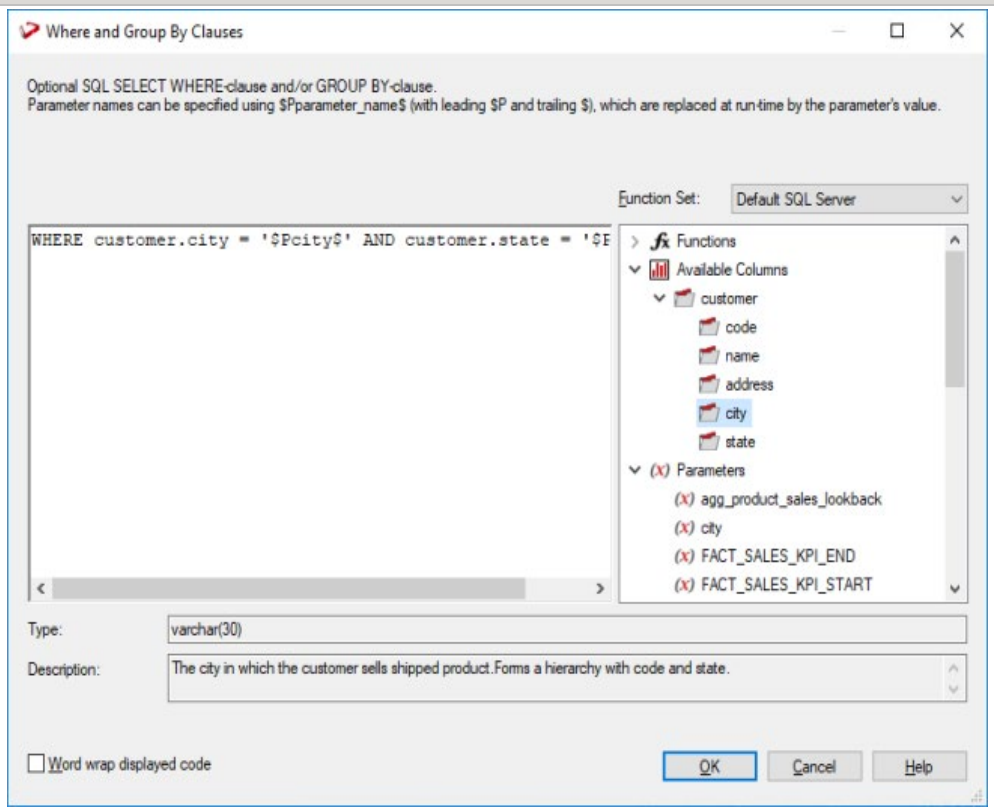
© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
Load Type	Method of loading data into the table. The available options are dependent on the **Source Connection**. Defaults to the 'Default Load Type' of the **Source Connection**. Can be specified via the Properties screen.
Source Connection	The connection that identifies the source database. Can be specified via the Properties screen.

General

Fields	Description
Select Distinct Values	Include the DISTINCT keyword in the SQL SELECT statement.
Source Schema	Schema within the source database where the source table resides.
Derive Source Tables(s) and Source Columns	Derive the **Source Table(s)** and **Source Column(s)** properties (of this screen) from the source details of this table's columns. **Note** The existing property values will be overwritten.
Source Table(s)	Name of the table or tables that the data is sourced from.
Override Source Column/Transformations	Ignore the source and transformation details of this table's columns and instead use the override details specified below.
Where and Group By Clauses	Optional SQL SELECT WHERE-clause and/or GROUP BY-clause. Parameter names can be specified using $Pparameter_name$ (with leading $P and trailing $), which are replaced at run-time by the parameter's value. **Tip** This is where you can build a statement to handle change data. Parameter values can be in-line replaced and included in the 'Where' clause. Prefix the parameter name with a '$P' and add a trailing '$'. For example, if we have a parameter called SALES_LOCATION_CODE we could construct a statement, such as WHERE location_code = '$PSALES_LOCATION_CODE$' AND region_code = 'NY'. When this statement is executed, the value of the parameter will replace the parameter name. For example if the parameter was set to 'New York' then the statement would execute as: WHERE location_code = 'New York' AND region_code = 'NY'. Clicking the SQL button opens the **Where and Group By Clauses** editor which provides control settings to assist you in constructing the SQL statement.

© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
	
Override Load SQL	Optional SQL statement to load data into the table, which overrides all other properties. The specified SQL will be executed instead of generated SQL. For a linked database specify a complete INSERT statement. For an ODBC source specify only the SELECT statement.
Source View Name	**Oracle only**: Name of the database view to be created by WhereScape RED (if necessary) in the source system that will be used to select the source data. Refer to **Remote View Extract** for details.

Native ODBC Load

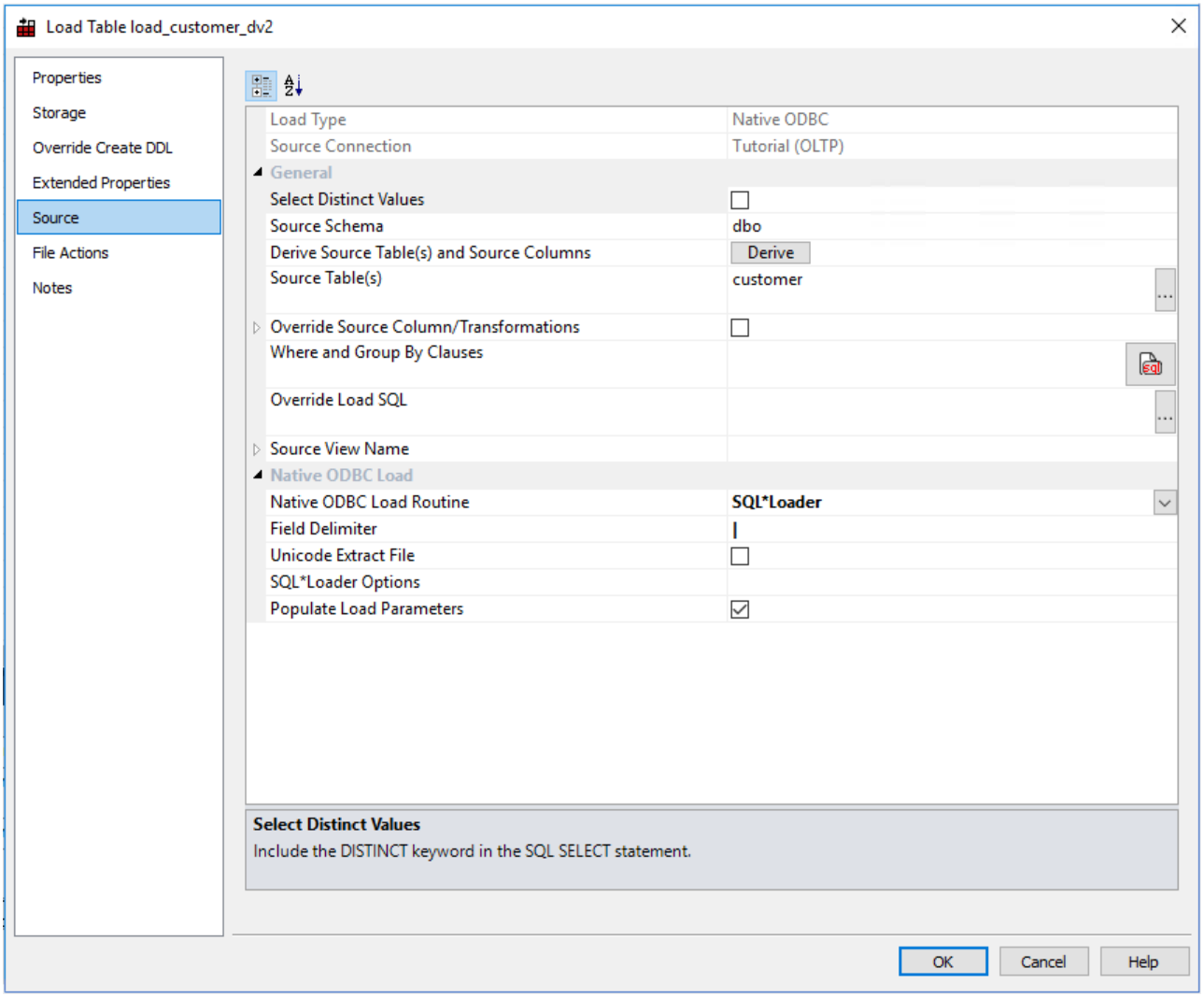
Fields	Description
Native ODBC Load Routine	File Loader utility/mechanism to use to load the generated extract file.
Field Delimiter	Character that separates the fields within each record of the generated extract file. The default value is a | character (pipe). This should be changed if pipes are possible in the source data.
Unicode Extract File	Data will be loaded via a Unicode format file. The default is unchecked and this field is not available FOR DB2.
SQL*Loader Options	**Oracle Only** - use this field to optionally specify an Oracle SQL*Loader OPTIONS clause and/or WHEN clause. Multiple options can be specified via the OPTIONS clause by separating each option with a comma. The property can be used to specify that SQL*Loader uses the efficient **Direct Path method** to load data into Oracle and/or a WHEN clause to selectively load rows based on a condition. Examples: OPTIONS(DIRECT=TRUE) specifies that SQL*Loader uses the Direct Path load method WHEN (3) = 'Y' specifies that only rows where column 3 contains the value Y are loaded

© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
	Note Use of the "not" symbol (!) in the WHEN statement is only supported for Linux/UNIX file loads. This symbol is not supported for Windows based loads.
BULK INSERT Maximum Errors	**SQL Server Only** - Maximum number of errors accepted before a load failure is triggered. The default for this option is 0.
BULK INSERT Additional Options	**SQL Server Only** - Optional comma-delimited list of options to control the behavior of the SQL Server BULK INSERT command. For example the following options will respectively lock the table; set the batch size to 100,000 rows; and fail the load when the first error occurs: TABLOCK, BATCHSIZE=100000.
Populate Load Parameters	Populate any load-related WhereScape RED parameters, which may be used for validation purposes.

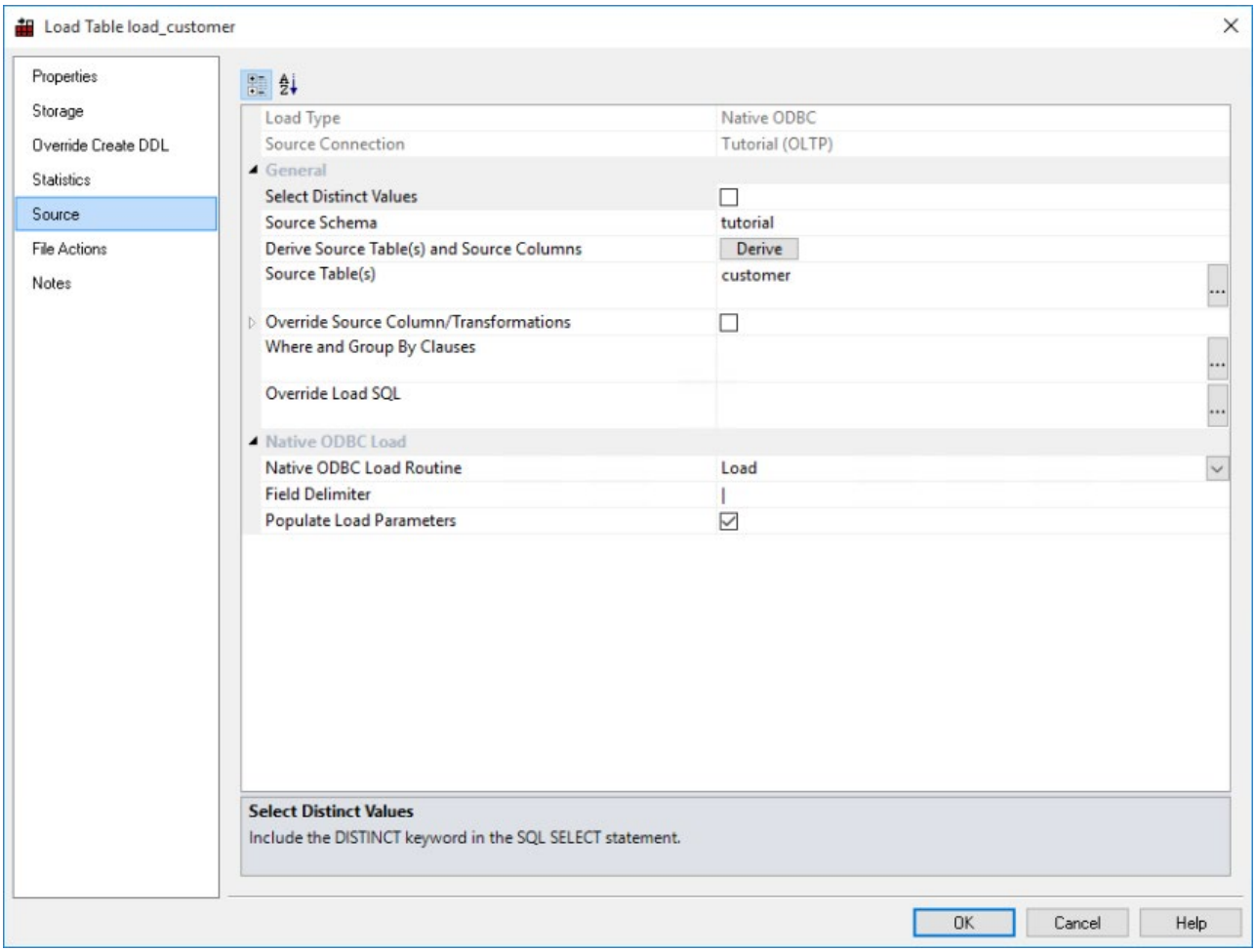
Oracle and DB2 Source screen examples:

Oracle Source Screen:



© Copyright, WhereScape Software Ltd All rights reserved.

DB2 Source Screen:



File Actions

A file action defines a copy, move or delete step that happens to **Native ODBC** temporary files after a load is run.

There are six different file action programs:

File Action	Description
copy all files	Copies all temporary files for the Load table (including the temporary data file) to another specified directory.
copy data files	Copies the **Native ODBC** Load table's temporary data file to another specified directory. Also enables a new file name to be specified.
delete all files	Deletes all temporary files for the Load table (including the temporary data file).
delete data files	Deletes the **Native ODBC** Load table's temporary data file.
move all files	Moves all temporary files for the Load table (including the temporary data file) to another specified directory.
move data files	Moves the **Native ODBC** Load table's temporary data file to another specified directory. Also enables a new file name to be specified.

Note

A Load table may have more than one file action. File actions are processed in order, based on the action number.

© Copyright, WhereScape Software Ltd All rights reserved.

Creating a File Action

To create a file action:

1. Click the **File Actions** tab of the Load table **Properties** window.
2. Click **Add new action** and then select **After load** from the **Action Type** drop-down list.
3. Enter the **Action Description**.
4. Select the **Action Program** from the drop-down list (see above for the options).
   - For action programs of copy... and move..., enter the target directory name.
   - For copy data files and move data files, optionally enter the file's new name.
5. Click **Save (Update) Action**.
6. Repeat if necessary for additional file actions.
7. Click **OK**.

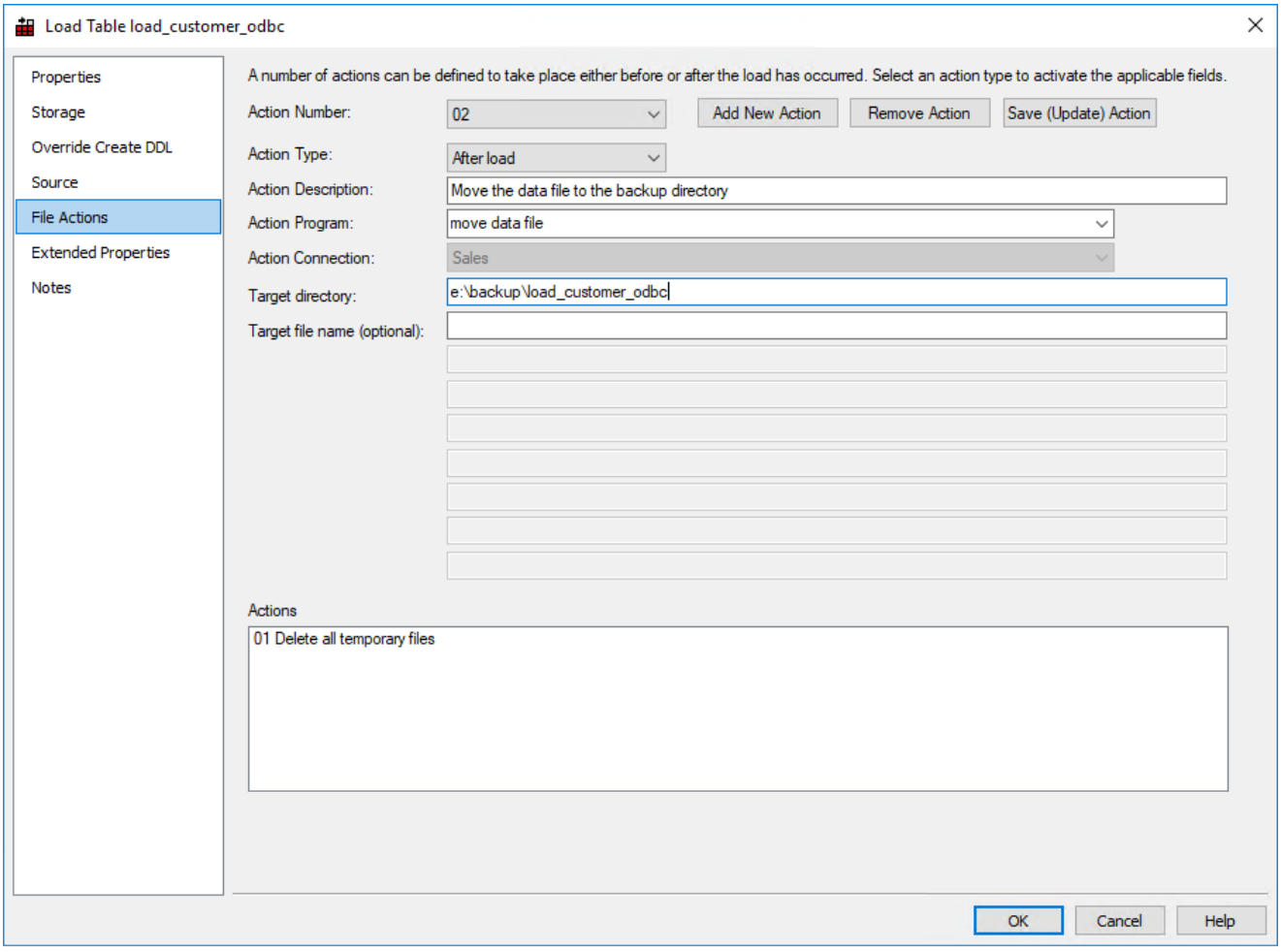
File Action Example

The following example moves the Native ODBC Load data file to another directory then deletes all temporary files.

Note

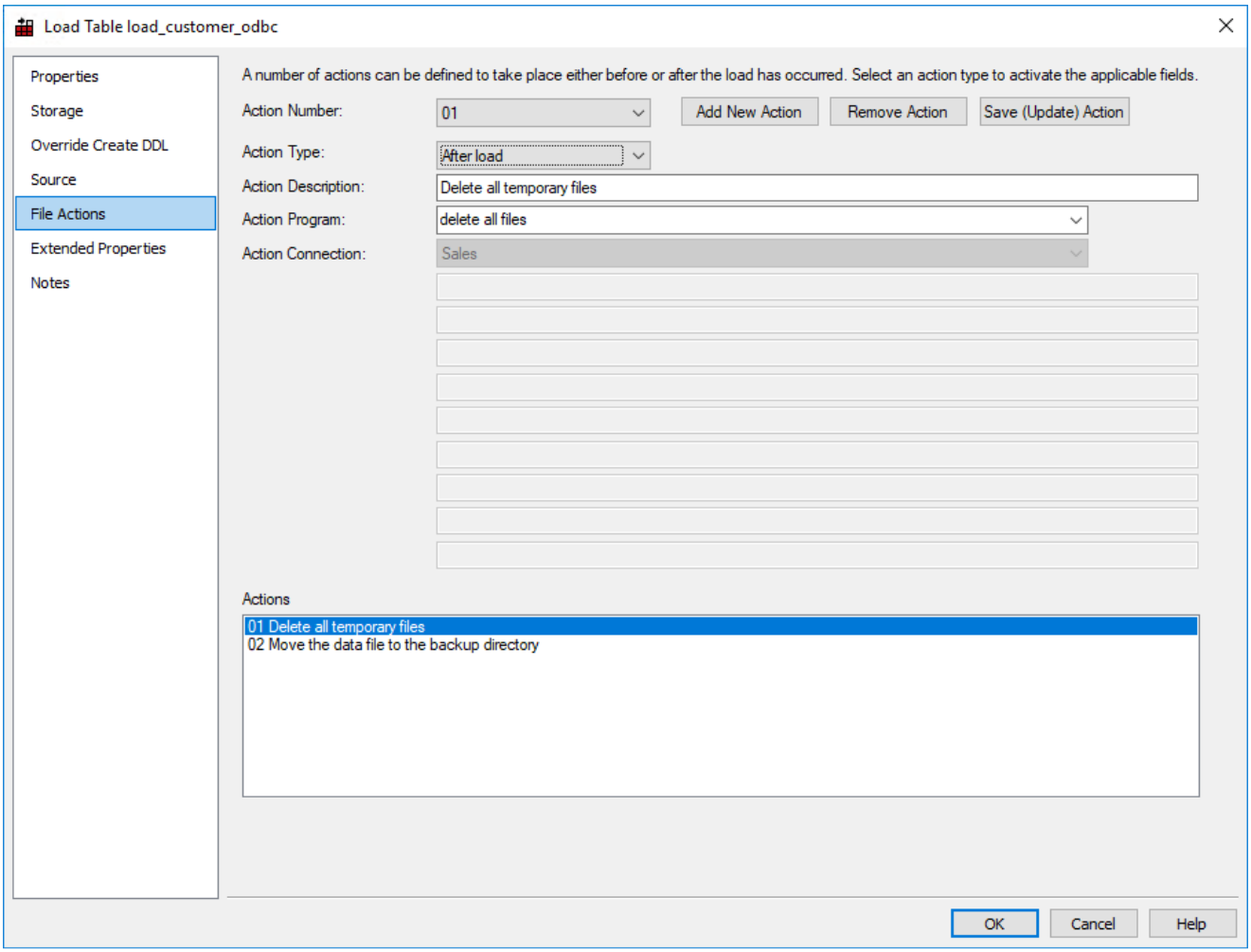
The data file is not deleted as it has been moved prior to the delete action.

Sample **move data file** action:



© Copyright, WhereScape Software Ltd All rights reserved.

Sample **delete all files** action:



© Copyright, WhereScape Software Ltd All rights reserved.

Flat File Loads

Flat file loads can be performed from either a **Windows**, **UNIX/Linux** or **Hadoop** connection. As with all other load types it is easier to use the drag and drop functionality to create load tables.

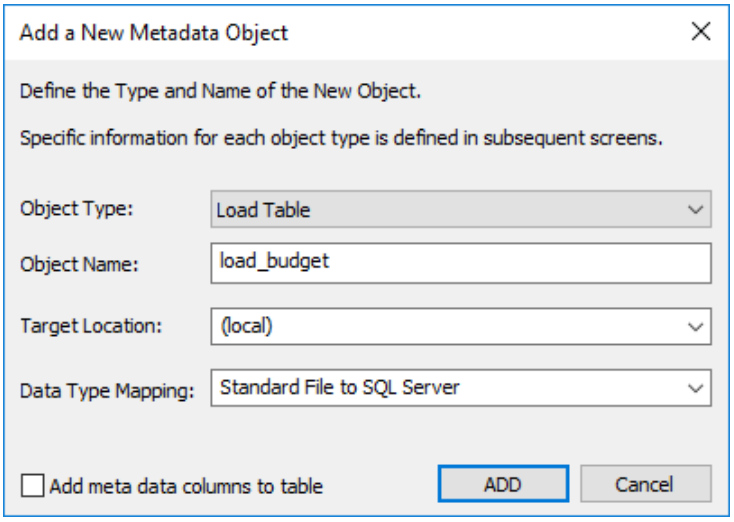
Flat files can also be loaded using SQL Server Integration Services (SSIS). For Flat File load instructions using SSIS, refer to the next section Loading Data from a Flat File using SSIS.

Note

Greenplum - To process Greenplum file loads via the Windows Scheduler, please ensure that the Windows Scheduler user has the all the necessary permissions including having the ssh host key saved.

The **drag and drop** process for flat files is as follows:

1. **Browse** to the directory and file via the appropriate **Connections**.
2. Double-click on the **Load Table** object in the left pane to create a drop target.
3. **Drag** the file from the right pane and **drop** it into the middle pane.
4. The dialog below appears. Rename the file if necessary and then click the **ADD** button.

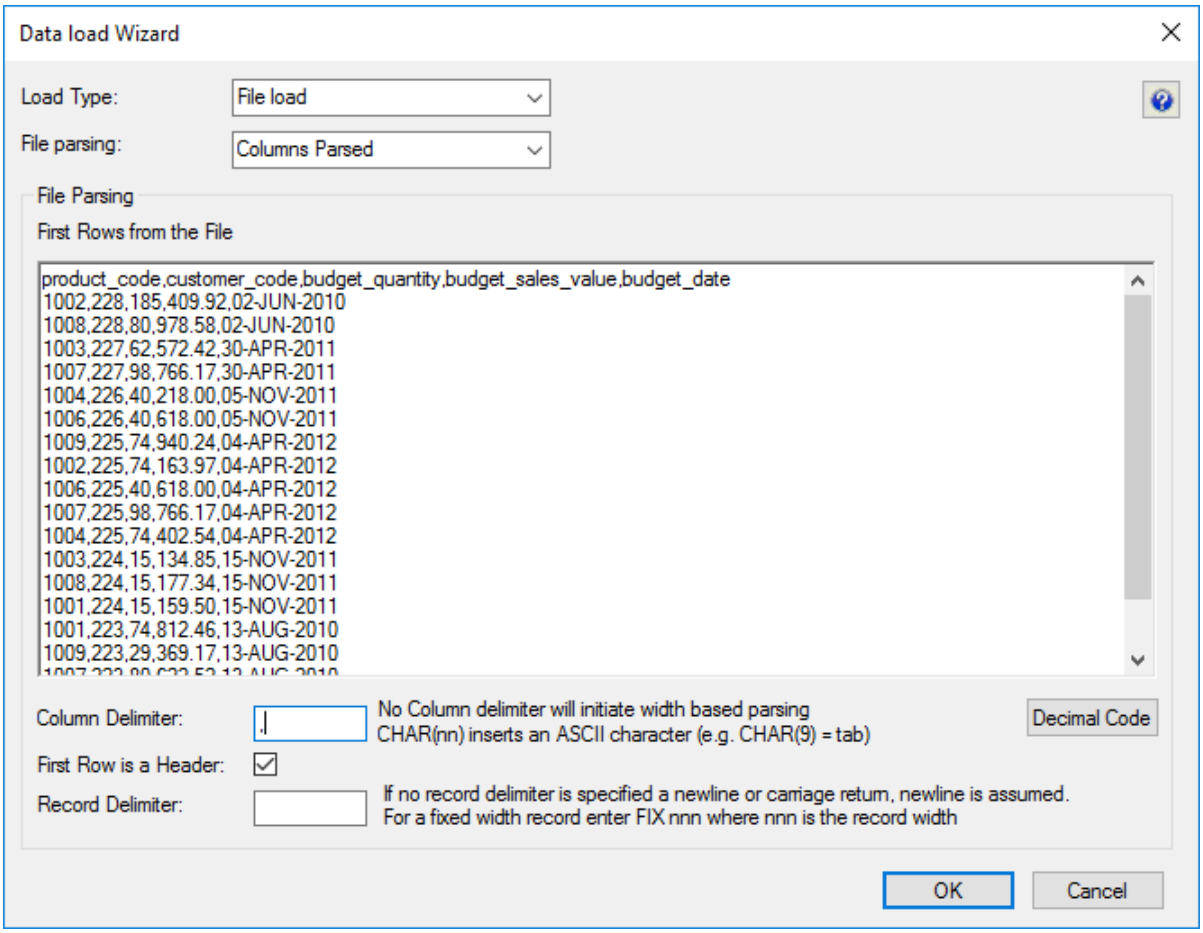


Note:

The option **Add meta data columns to table** is used for creating Load tables that are used in creating Data Vault objects. If this option is selected, two DSS columns (**dss_record_source** and **dss_load_date**) are included in the meta data for the table and are populated by transformations. These two DSS columns could equally be applied to other Load tables not used in a Data Vault system but are particularly important to comply with the Data Vault standards. Refer to **Data Vaults** for details.

5. The following dialog displays for file loads from **Windows, UNIX/Linux** and **Hadoop** connections.

© Copyright, WhereScape Software Ltd All rights reserved.

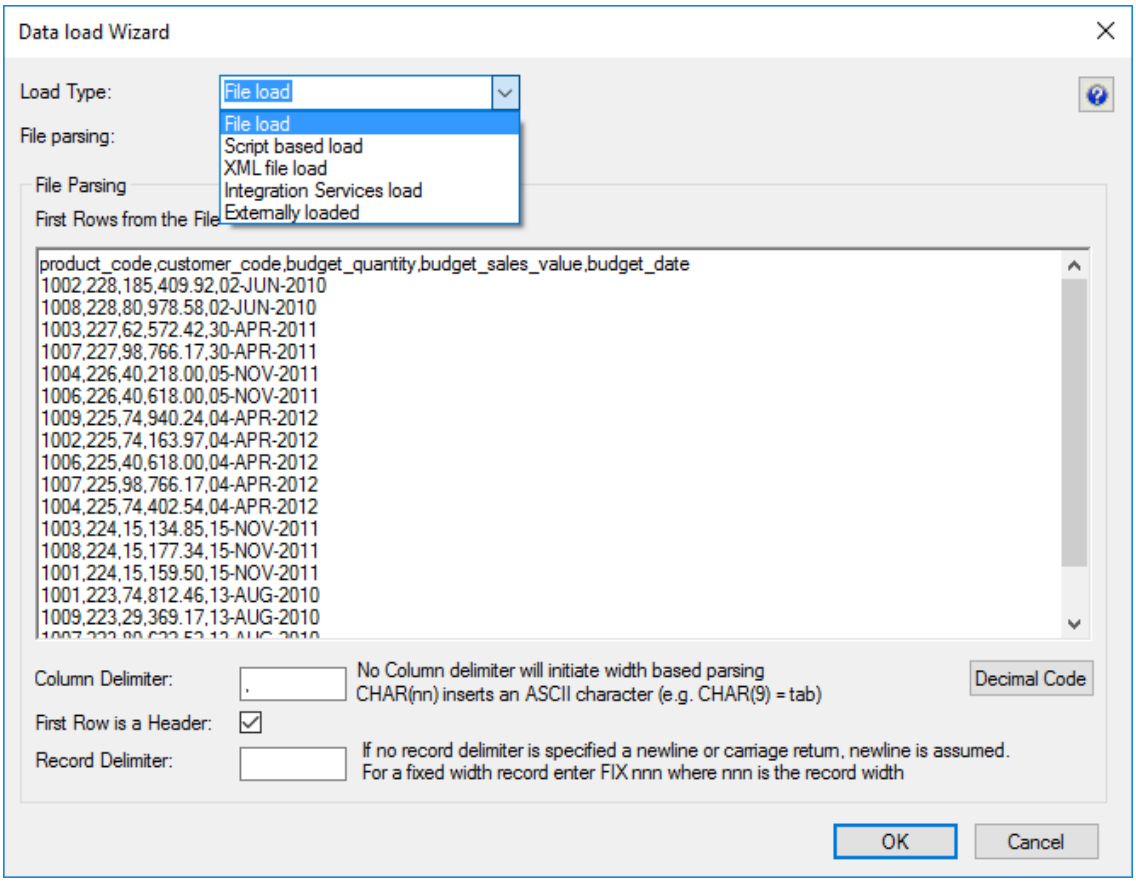


Notes:

- **First Row is a Header**- To make changes in relational database tables after a table has been defined, users can edit the **First Row is a Header** option in the **Source** tab of the relevant table.
- **Hive** - For File loads into Hive tables, the **First Row is a Header** option is a table option. Please ensure this option is selected in the file load wizard if you want to load files with header rows.
  To make any changes after Hive tables have been defined, the **First Row is a Header** option can be found in the **Storage** tab of the relevant Hive table.

6. The load type selected in the **New Table Default Load Type** field in the connection Properties screen is the pre-selected option in the **Load type** drop-down list.
   - To change the desired load type and file parsing, use the **Load type** and **File parsing** drop-down list options.

© Copyright, WhereScape Software Ltd All rights reserved.



Load type options

- The **File based load** options results in a load where the bulk of the load management is handled by the scheduler.
- The **Script based load** option will make WhereScape RED generate a host script and the load table will be a **Script based load**. This host script is executed by the scheduler to perform the load. For further details about Script based loads, refer to **Script based load**.
- The **XML file load** option will only be an available load option from a Windows connection. To see more details about specific XML loads, refer to **XML File load**.
- The **Integration Services load** option will load the file via an Integration Services Package that is generated and executed dynamically at run time. This option is only available from a Windows connection. For specific details about this load option, refer to Loading Data from a Flat File using SSIS.
- The **Externally loaded** option will not execute an actual load into the table but will process the actions specified in the Post Load procedure property. Any **After** transformations recorded against any of the columns in an Externally loaded table will also be processed.

Post load procedures can either be manually generated from a RED provided procedure outline or generated leveraging a RED template—refer to **Rebuilding Update Procedures** for details.

Note:

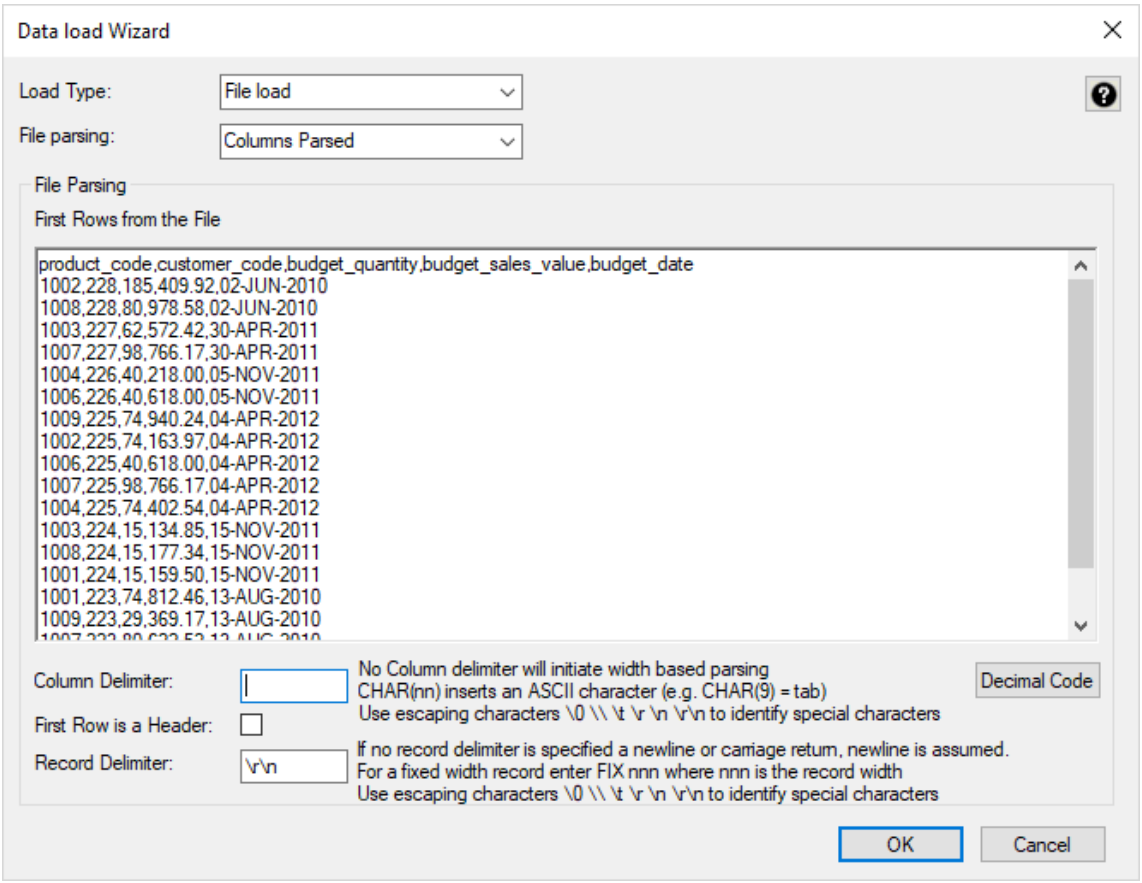
For custom database targets, the option to use Post Load Scripts is provided as an alternative to post load procedure. Post load scripts can be generated using a RED template—refer to **Rebuilding Update Procedures** for details

File parsing options

- **Single data column** - with this option, the majority of the work in terms of parsing the file must occur in a subsequent procedure within the data warehouse. The data is dumped into a single column. The task of coding a procedure to parse the data must then be undertaken. Three columns are created under Oracle. These include the data column, a sequence column (row_sequence) and the file name column (row_file_name). The file name and sequence columns can be deleted if they are not required for a File based load.

© Copyright, WhereScape Software Ltd All rights reserved.

- **Columns parsed** - with this option, WhereScape RED will attempt to parse the columns. You will be asked for details and for the column delimiter. You then step through the columns providing names and data types. WhereScape RED attempts to guess the data type, but it needs to be checked and the field length will probably need to be adjusted.



Notes:

- **DB2** - DB2 databases do not support data import from files with header row.
- The **Decimal Code** button will show the decimal value of each character in the lines retrieved from the source file. These decimal codes will be shown below each line and are green.

7. Once the selection on the screen above is completed, a screen displays which enable the breakdown of the source data into columns.
   If no delimiter is entered, then width based parsing is assumed and an addition width size is prompted for.

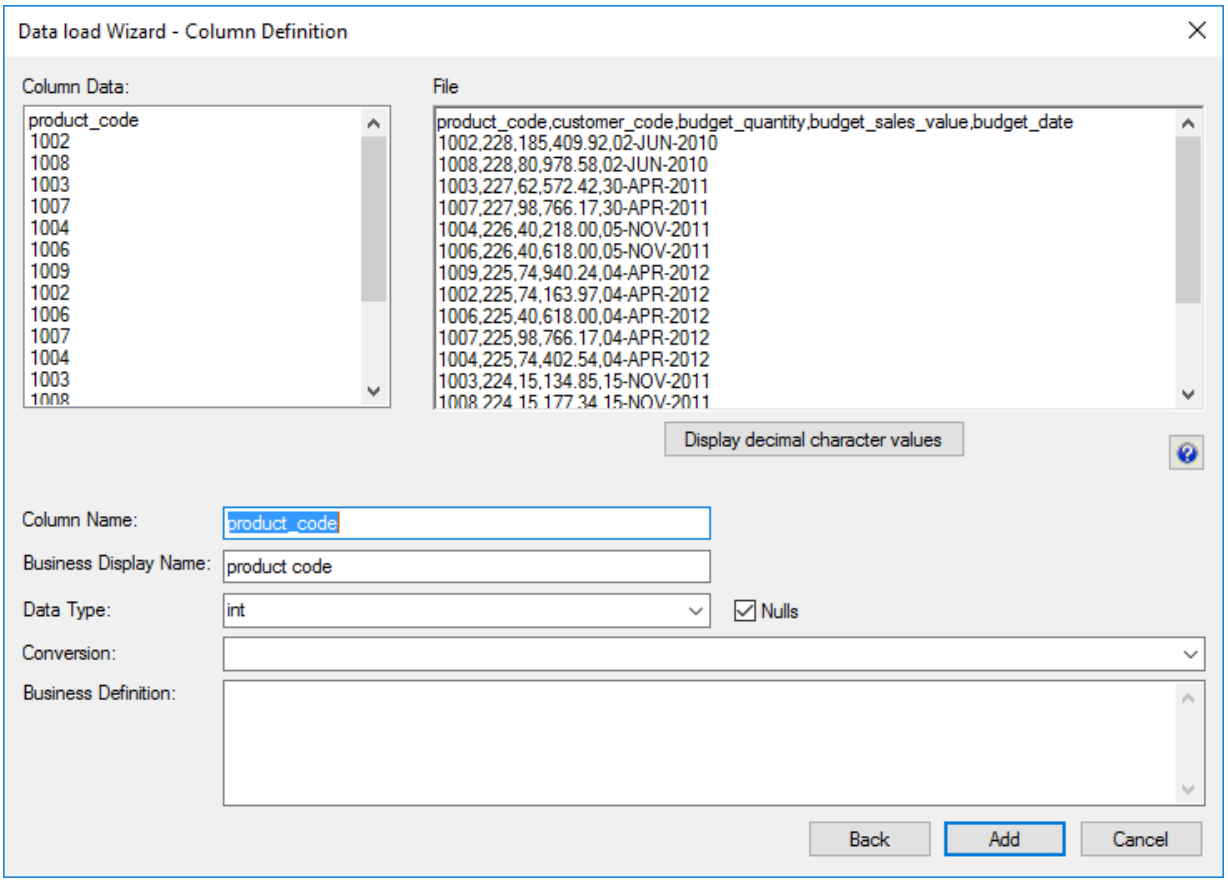
Notes:

- If no delimiter is entered, then **width based parsing** is assumed and an addition width size is prompted.
- Teradata - If this is a **Fixed Width** file loaded via TPT, the source file format can be specified later in the **File Load Source Screen** of the Load table's **Properties** window.

© Copyright, WhereScape Software Ltd All rights reserved.

The following screen is an example of the file parsing technique.

Use the **Back** button to revert to the previous column if an incorrect width or delimiter is entered.



Conversion

Oracle

During the parsing of the columns an Oracle SQL*Loader conversion string can be used and is required in the case of dates for Oracle. Refer to the Oracle Sql loader manual for syntax.

Any Oracle function can be used if enclosed in quotes and with the column name prefixed by a colon. For example: when loading a column called 'product_name', use the following syntax to bring over only the first 30 characters:

```
product_name "substr(:product_name,1,30)"
```

A special variable %FILE_NAME% can be used in an Oracle File based load. This will be substituted with the actual source file name.

To load the full file name (including the path) into a field, enter this transformation:

```
"RTRIM('%FILE_NAME%')"
```

To remove the path from the full file name on a UNIX or Linux system, enter this transformation:

```
"SUBSTR('%FILE_NAME%',INSTR('%FILE_NAME%','/',-1,1)+1)"
```

© Copyright, WhereScape Software Ltd All rights reserved.

SQL

After Load transformations can be processed. The conversion string can be entered in the relevant column conversion field during the parsing of the columns, however, to process the column conversion, users will need to do the following after the table is created and loaded:

1. Go into the **Properties** of the loaded table's relevant column(s) by double-clicking on the column(s) in the middle pane.
2. Click the **Transformation** tab.
3. Select the **After load** option in the **Transformation Stage** drop-down list.
4. **Recreate** the load table.

Note

If running from the Windows scheduler, then a wildcard must be used in the file name if you wish to use the %FILE_NAME% variable. If a wild card is not used, then the literal file name appears enclosed in double quotes. When a wild card is used, Windows looks up the file and returns the actual file name with no enclosing double quotes.

Understanding SQL*Loader

When using either type of loading method **SQL*Loader** is called to perform the actual load. There is therefore a requirement that the executable called **'sqlldr'** be in the path.

Under **UNIX** this means that the UNIX user must be able to execute 'sqlldr' from the command prompt under a normal log on.

Under **Windows** 'sqlldr' must execute when the command is issued from a DOS prompt.

In both the Windows and UNIX environments sqlldr returns a result code indicating if the load worked fully, partially or failed. WhereScape RED makes use of this returned code to report on the results of the load.

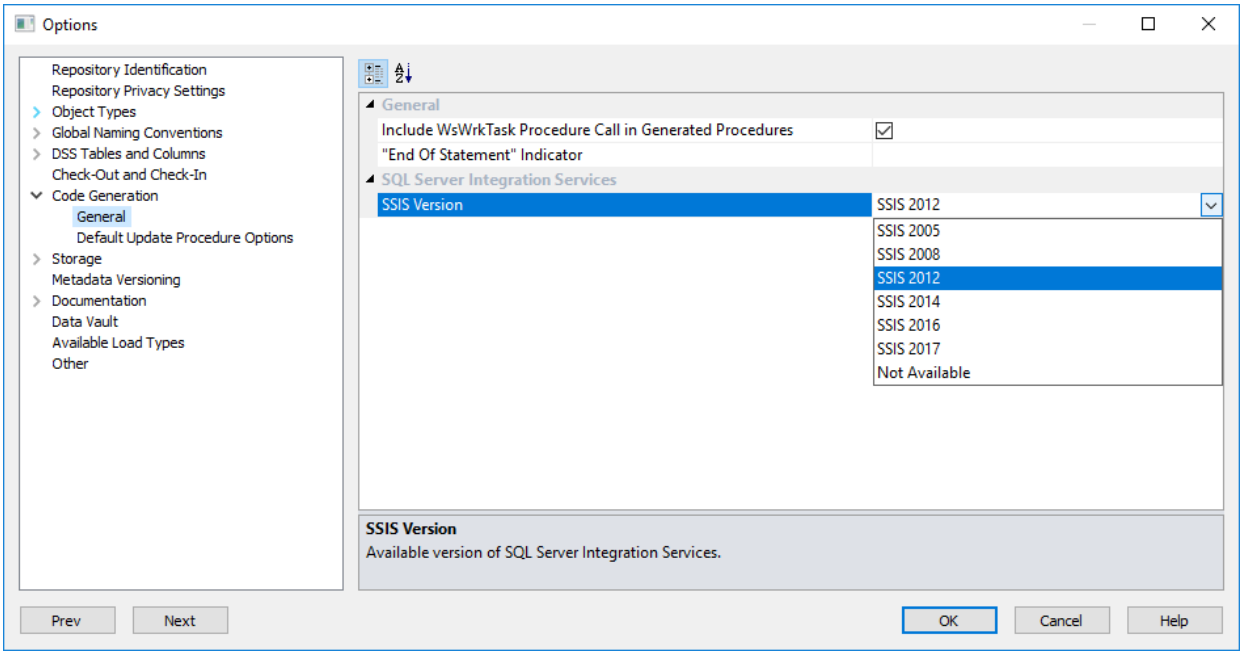
© Copyright, WhereScape Software Ltd All rights reserved.

Loading Data from Flat Files using SSIS

Flat files can be loaded into RED from a **Windows** Connection using SQL Server Integration Services (SSIS).

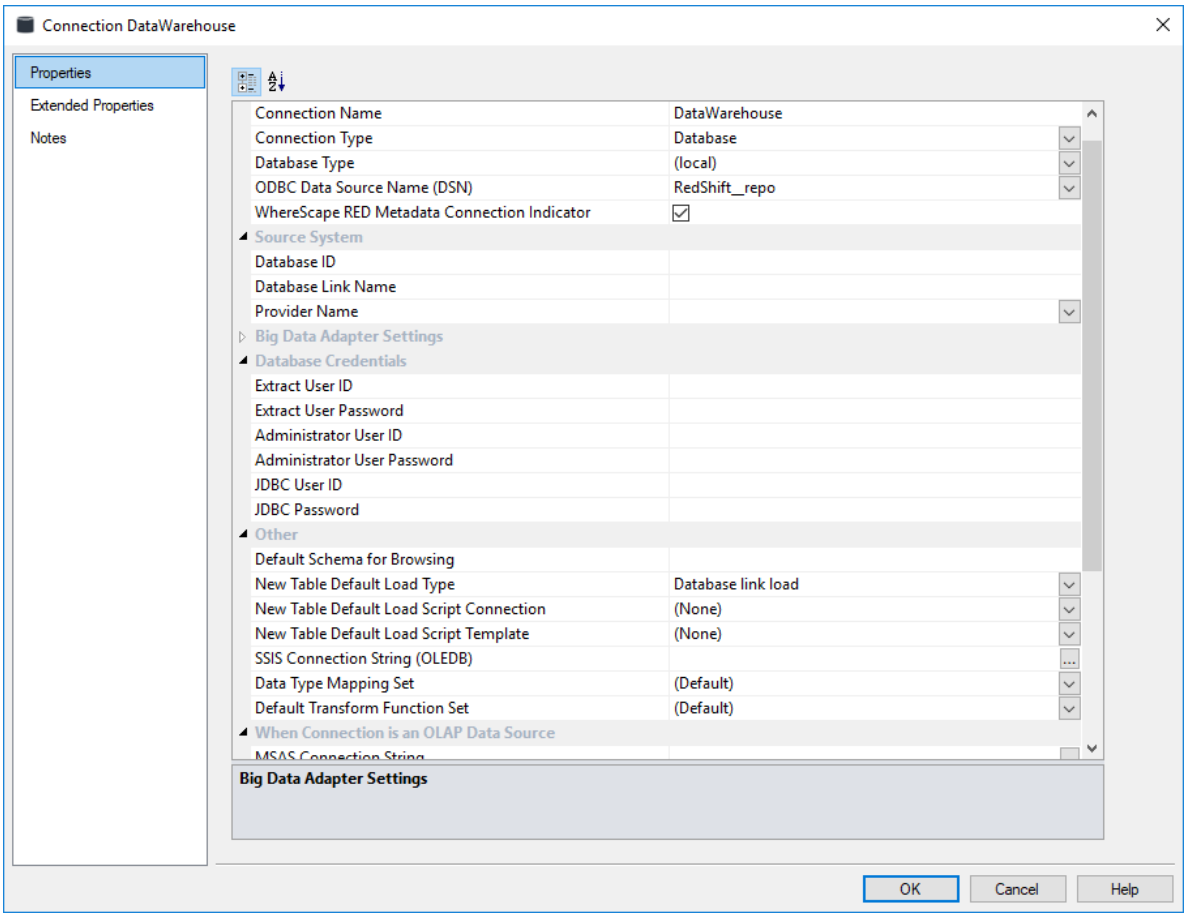
The instructions below detail how to add the SSIS connection string to the data warehouse connection and load flat files using the drag and drop functionality to create Load tables. To load files via SSIS, the SSIS connection string must be defined in the Data Warehouse connection.

To use SSIS loading, ensure that SSIS loads are enabled and that the SSIS version is set correctly in **Tools > Options > Code Generation > General**.

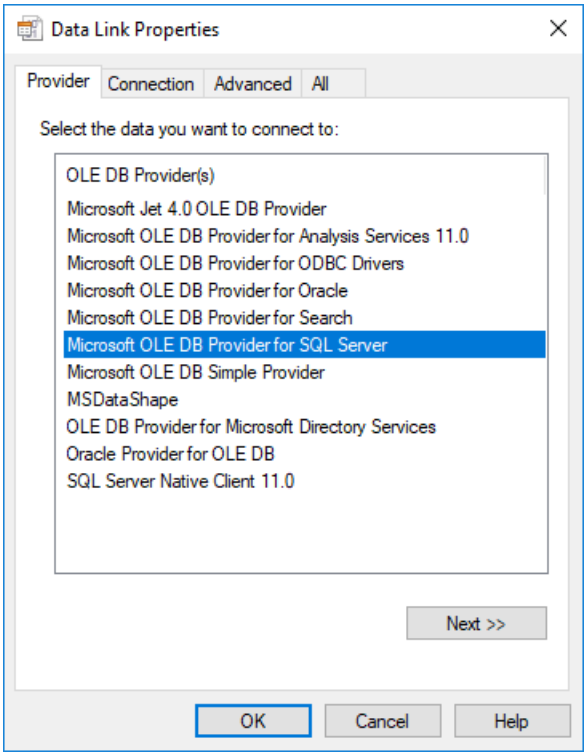


1. To load files via SSIS, the SSIS connection string must be defined in the DataWarehouse connection for the **Destination** connection to be specified:
    - Double-click the **DataWarehouse** connection in the object explorer to open up the **Properties** window.
    - Click the ellipsis button to open the wizard and define the SSIS connection string.

© Copyright, WhereScape Software Ltd All rights reserved.



2. On the Provider tab, select the relevant **OLE DB Provider** and click **Next**.

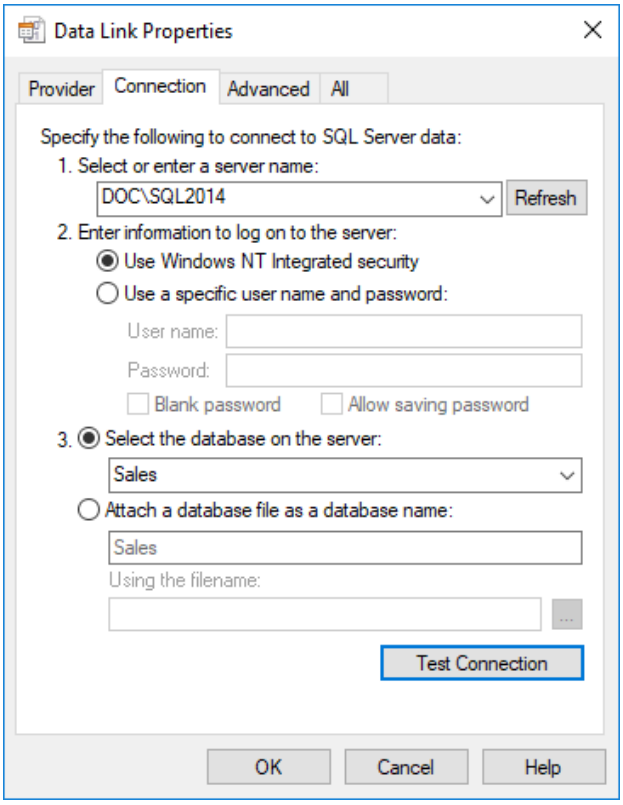


© Copyright, WhereScape Software Ltd All rights reserved.

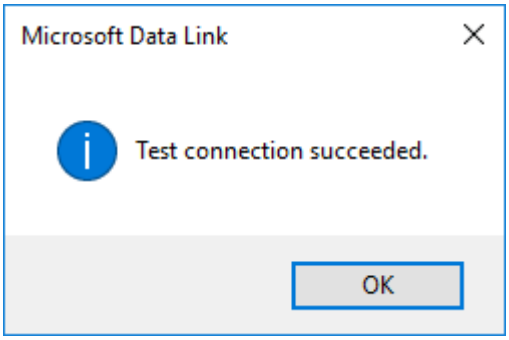
3. On the **Connection** tab, select the **server name**, enter the information to log in to the server and select the **database** on the server. Click **Test Connection**.

Note:

When using a specific **user name** and **password** to connect to the server instead of using Windows integrated security, the **Allow saving password** check-box must be selected. It is also recommended that the password on the SSIS connection string field that is displayed in the connection properties is replaced with the **$PASSWORD$** token that is substituted at runtime.

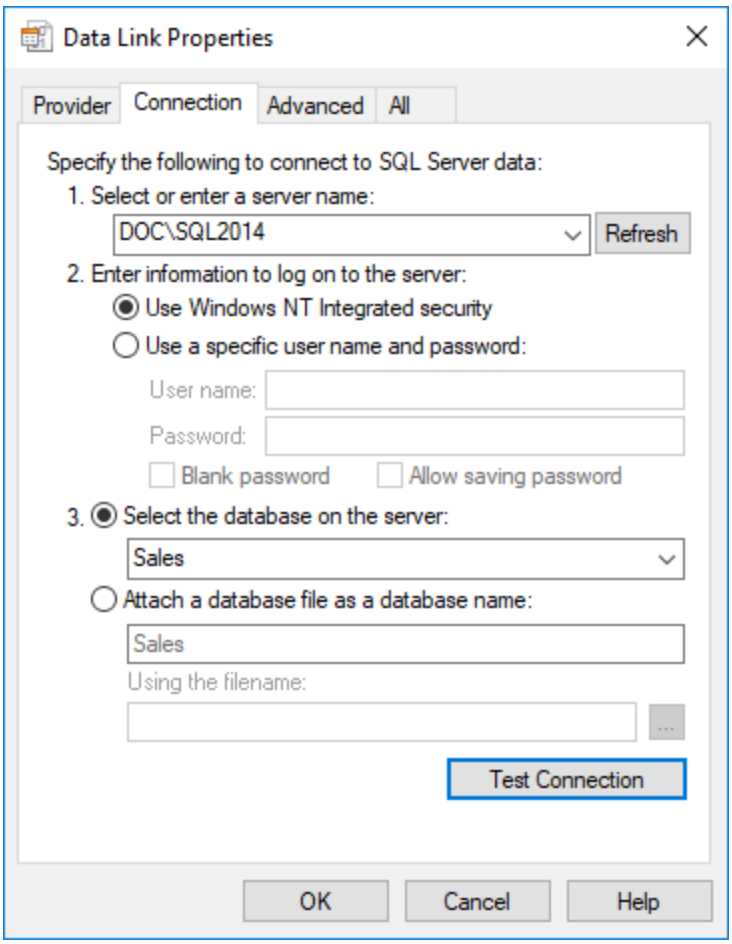


4. Click **OK**.

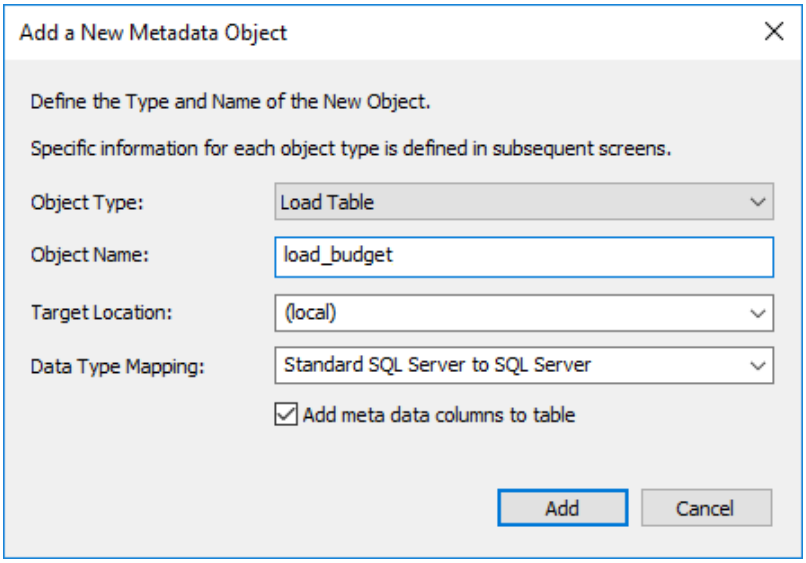


© Copyright, WhereScape Software Ltd All rights reserved.

5. Click **OK** on the **Data Link Properties** dialog to save the SSIS connection string settings.

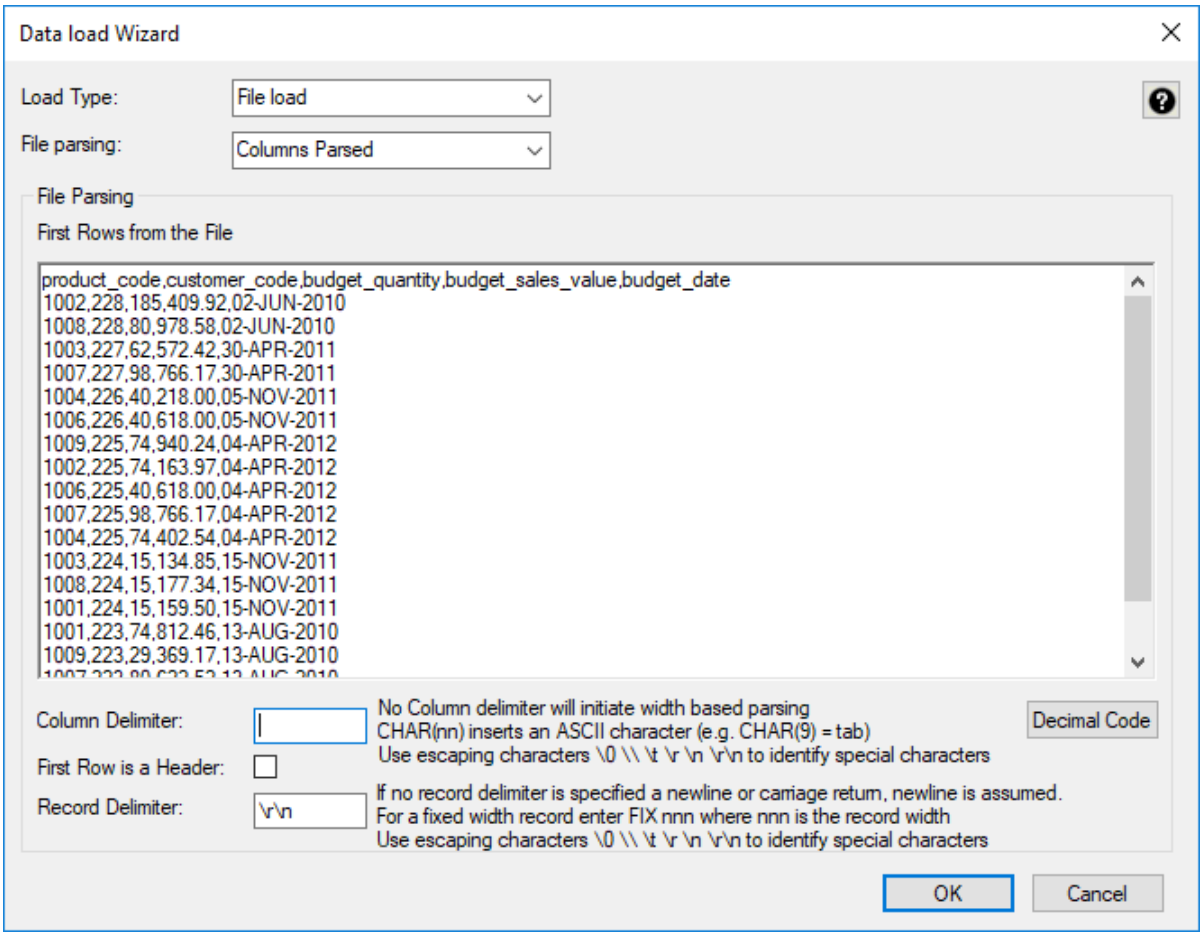


6. Click **OK** on the Data Warehouse connection properties screen.
7. Browse to the directory and file from the **Windows** connection.
8. Double-click on the **Load Table** object in the left pane to create a drop target.
9. Drag the file from the right pane and drop it into the middle pane. The window below appears.



© Copyright, WhereScape Software Ltd All rights reserved.

10. Click the **ADD** button, the following screen appears:



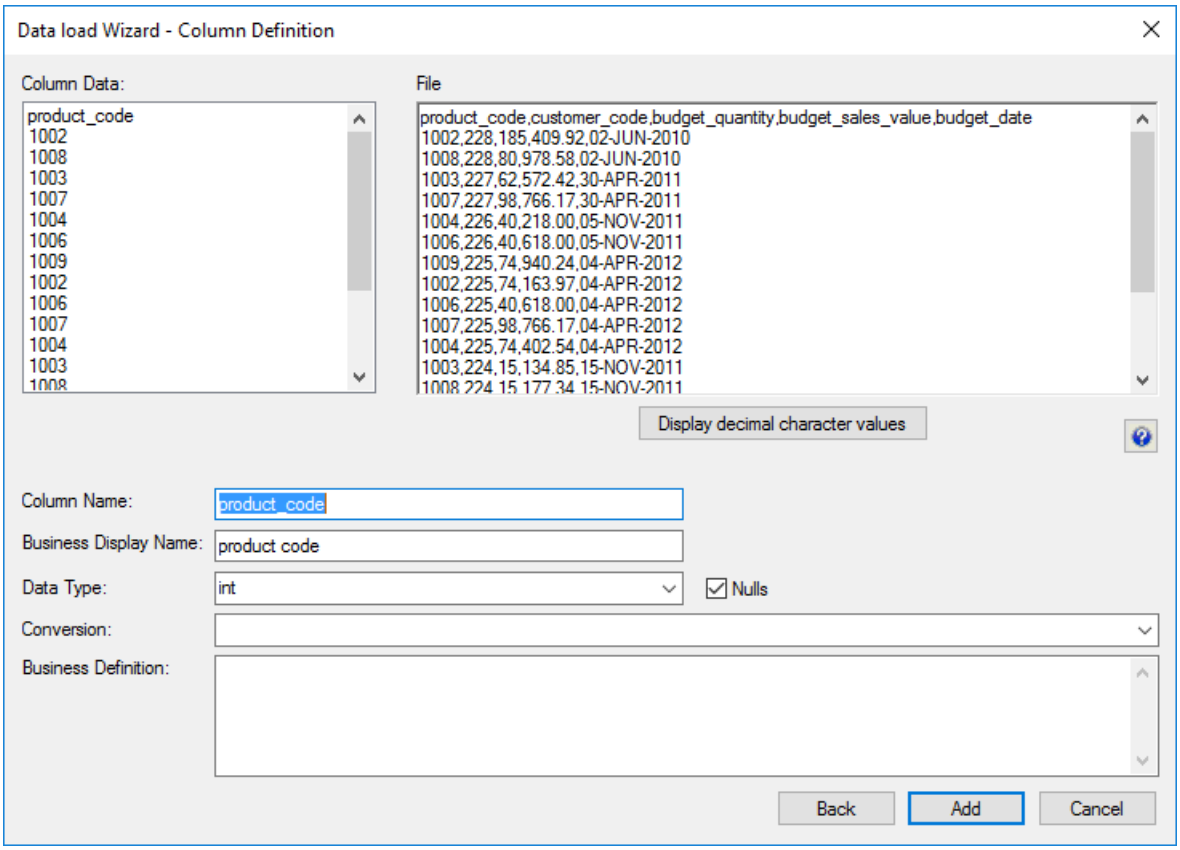
11. Configure the settings and click **OK** to continue. See the previous section for details about these settings.

Notes:

- DB2 databases do not support data import from files with header row.
- The **Decimal Code** button shows the decimal value of each character in the lines retrieved from the source file. These decimal codes are shown in green below each line.

© Copyright, WhereScape Software Ltd All rights reserved.

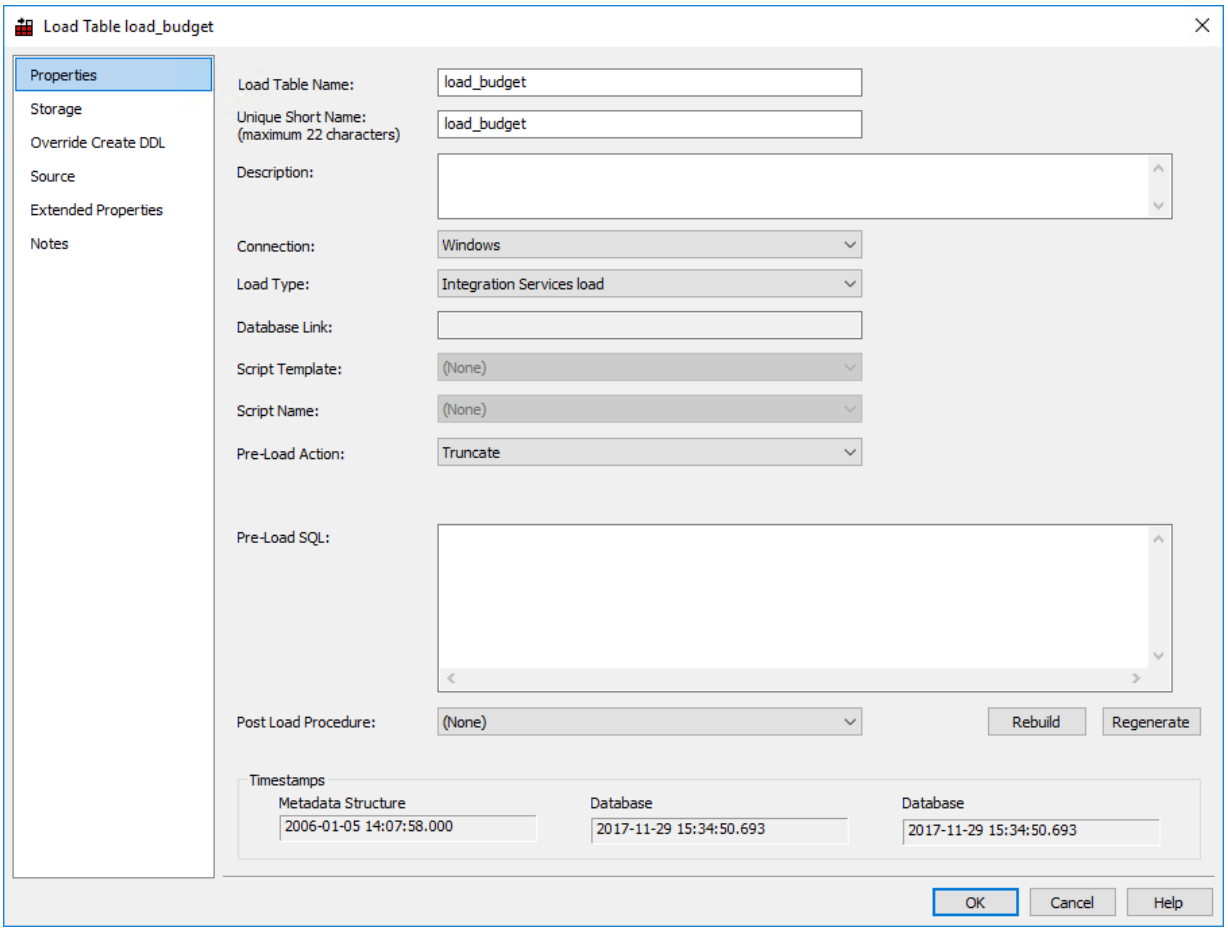
12. The next screen enables you to breakdown the source data into columns. If no delimiter is entered then width based parsing is assumed and an addition width size is prompted. The following screen is an example of the file parsing technique.



Use the **Back** button to revert to the previous column if an incorrect width or delimiter is entered.

© Copyright, WhereScape Software Ltd All rights reserved.

13. On the **Properties** screen for the new load table, select **Integration Services Load** as the Load Type. Click **OK**.
This will create and execute a SSIS package at run time to load data into the data warehouse load table.



Notes:

- If the table is changed to an Integration Services load and has been set up using the wizard for the "File load (columns parsed)" flow, some columns might have transformations set up that will not work.
- In RED 6.8.3.4 date/time fields have transformations that are invalid for SSIS and will make the load fail. Since SSIS does not provide any configuration for the parsing of date/time fields, if users have any date/time field special requirements, file or script-based loads provide are a better load option instead.

14. Click **Yes** to Create and Load the table.

© Copyright, WhereScape Software Ltd All rights reserved.

Flat File Load - Source Screen

The fields for the **Flat File Source Screen** are described below. See example source screens in **File and Script based load Source screens**.

Tip

If the file has been **dragged and dropped** into the load table (middle pane) then some of the fields on this tab are automatically populated.

Fields	Description
Load Type	Method of loading data into the table. The available options are dependent on the **Source Connection**. Defaults to the 'Default Load Type' of the **Source Connection**. Can be specified via the Properties screen.
Source Connection	Connection to the data source (database or file system). Can be specified via the **Properties** screen.
Load Script Template	Available for script loads only and only if there is a valid template available. Select the template to use when generating a load script, or select (None) to use RED's built-in load script generator. Only templates with the correct Type and Target DB will appear in this drop-down list. Refer to **Templates** for details.
Source File Details	Source File identification and definition information.
Source Directory	The full path (absolute path) of the folder/directory containing the Source File on the Windows or UNIX/Linux system.
Source File Name	The name of the source file containing the data to be loaded.
Source File Character Encoding	**PDW option only**- The character encoding of the input datafile for PDW file loads. Select between **ASCII**, **UTF8**, **UTF16** or **UTF16BE**. The **UTF8** encoding cannot be specified for a fixed width source file (i.e. when the Source File Field Delimiter is blank) because UTF8 is a variable length character encoding.
Source File Field Delimiter	Optional character that separates the fields within each record of the Source File. The delimiter identifies the end of each field. Common field delimiters are tab, comma, colon, semi-colon, pipe, tilde. If no field delimiter is specified the record is regarded as fixed-width. **Note** If an ASCII character value is used this field may show as an unprintable character. To enter a special character enter the uppercase string CHAR with the ASCII value in brackets (e.g. CHAR(9)).
Source File Field Enclosure Delimiter	• **Oracle and PDW Only** - Optional character that delimits BOTH the start & end of a field value i.e. encapsulates a value. A double-quote(") is a common enclosure delimiter. WhereScape RED allows the specification of one enclosure delimiter that is used for both the start & end delimiters. Everything between the start & end enclosure delimiters is considered part of the field value. Enclosing a field's value is OPTIONAL but it must either be enclosed by BOTH a start and an end enclosure delimiter or BOTH of them must be absent. A field value can contain the enclosure delimiter character as embedded data if each embedded instance is "escaped" by the same character. • **PDW Only** - For **PDW** loads, the delimiter can be specified either as a character or as a hex value (such as 0x22 for a double quote).

© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
Source File Record Terminator	Optional string to identify how each line/record in the Source File is ended/terminated/delineated. The system default is used when not specified. On UNIX/Linux systems, end-of-line is typically line-feed (ASCII 10). On Windows systems, end-of-line is typically carriage-return (ASCII 13) and line-feed (ASCII 10).
Source File has Field Headings/Labels	Indicates whether the first line of the Source File contains a heading/label for each field, which is not regarded as data so it should not be loaded.
Trigger File Details	Optional Trigger File identification and definition information. If a Trigger File is specified, the Source File will not be loaded until the Trigger File is available.
Trigger File Path	The purpose of the trigger file is to indicate that the copying/loading of the main file has completed and that it is now safe to load the file. Secondly the trigger file may contain control sums to validate the contents of the main load file. This field should contain the full path name to the directory in which a trigger file is located on the Windows or UNIX systems. If this field and/or the **Trigger Name** field is populated then the scheduler will look for this file rather than the actual load file.
Trigger File Name	Refers to the name of the file that is used as a trigger to indicate that the main load file is available. A trigger file typically contains check sums. If the trigger file exists then this is the file that is used in the Wait Seconds, and Action on wait expire processing. (see notes under Trigger Path above).
Trigger File Field Delimiter	If the trigger file provides control information then the delimiter identifies the field separation, or **\n** for a return. The data found will be loaded into parameter values whose names will be prefixed by the prefix specified and numbered **0** to **n**.
Trigger Parameter Name Prefix	If a trigger file and delimiter have been specified then the contents of the trigger file are loaded into parameters. The parameters will be prefixed by the contents of this field and suffixed by an underscore and the parameter number. These parameters can be viewed under **Tools > Parameters** from the WhereScape RED menu bar. The checking of these parameters can be achieved by generating a **Post Load procedure**. An example set of parameters may be budget_0, budget_1 and budget_2 where there are 3 values in the trigger file and the prefix is set to 'budget'.
Load Configuration	Configuration settings to control the load processing.
Check existence of Source File	This field controls whether the load process checks if the file exists before performing the load. This check can only be disabled when doing script-based TPT Loads from Windows and UNIX/Linux connections, which can improve the use of built-in TPT functionality to wait for the arrival of the file.
Wait for Source File or Trigger File	Controls whether the load process waits for the file to arrive when it is NOT available to load. Enabling this allows the wait-related properties to be specified. When a Trigger File is specified the load waits for it rather than the Source File.
Wait Limit (in seconds)	Maximum duration to wait when no file is available, which is specified in seconds e.g. 1800 seconds to wait up to 30 minutes. A value of 0 equates to no wait. If the wait time expires and the specified file cannot be found, then the load will exit with the status defined in Action, e.g. **Default Action = Error**
Action when Wait Limit Reached	Action to take when the Wait Limit has been reached and no file is available. The specified action impacts any remaining tasks in the currently running job. When 'Success' or 'Warning' is specified, the

© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
	WhereScape RED Scheduler will continue to run any dependent tasks in the running job. In contrast, specifying the 'Error' or 'Fatal Error' actions will cause the scheduler to stop/fail the job and hold any remaining tasks when the "Wait Limit" is reached.
BULK INSERT Options	**SQL Server only**- Optional comma-delimited list of options to control the behavior of the SQL Server BULK INSERT command. For example the following options will respectively lock the table; set the batch size to 100,000 rows; and fail the load when the first error occurs: TABLOCK, BATCHSIZE= 100000, MAXERRORS=0.
SQL*Loader Options	**Oracle only** - Use this field to optionally specify an Oracle SQL*Loader OPTIONS clause and/or WHEN clause. Multiple options can be specified via the OPTIONS clause by separating each option with a comma. The property can be used to specify that SQL*Loader uses the efficient **Direct Path method** to load data into Oracle and/or a WHEN clause to selectively load rows based on a condition. Examples: OPTIONS(DIRECT=TRUE) specifies that SQL*Loader uses the Direct Path load method WHEN (3) = 'Y' specifies that only rows where column 3 contains the value Y are loaded **Note** Use of the "not" symbol (!) in the WHEN statement is only supported for Linux/UNIX file loads. This symbol is not supported for Windows based loads.
Hadoop Loader	**Oracle only** - This option is only available for Script based loads from a Hadoop connection to Oracle. It allows selecting which Hadoop loader RED will use for loading the source file from Hadoop. • Select one of the two Big Data Connectors, Oracle Loader for Hadoop or Oracle SQL Connector for HDFS.
Character Set	**Oracle only** - Oracle compliant Character Name Set that tells SQL*Loader the character set of the input datafile.
DB2 Load Command Parameters	**DB2 only** - Optional Command Parameter Names/Keywords to control the behavior of the DB2 LOAD command/utility. For example, NONRECOVERABLE specifies that the load transaction is to be marked as un-recoverable and it will not be possible to recover it by a subsequent roll-forward action.
DB2 Load Command File Type Modifiers	**DB2 only** - Optional File Type Modifiers to control the behavior of the DB2 LOAD command/utility. For example, TIMESTAMPFORMAT="YYYY-MM-DD HH:MM:SS.UUUUUU"
Archived File Details	Optional Archived file identification and definition information. Once a file has been successfully loaded or processed, it can be optionally archived by moving it and/or renaming it.
Compress Source File when Archive	Optionally compresses the successfully loaded Source File if it is archived.
Archived Source File Path	Optional full path (absolute path) of the folder/directory to MOVE the successfully loaded Source File on the Windows or UNIX/Linux system.
Archived Source File Name	Optional new name to RENAME the successfully loaded Source File to. By default, the original file name is used it is optional to rename it. However, the in-built variable $SEQUENCE$ can be used to include a unique sequence number in the new name. Likewise, the in-built variables $YYYY$, MM, DD, HH, MI and/or SS can be used to include the number of the current year, month, day, hour, minute and/or second respectively. The date/time components can be used

© Copyright, WhereScape Software Ltd All rights reserved.

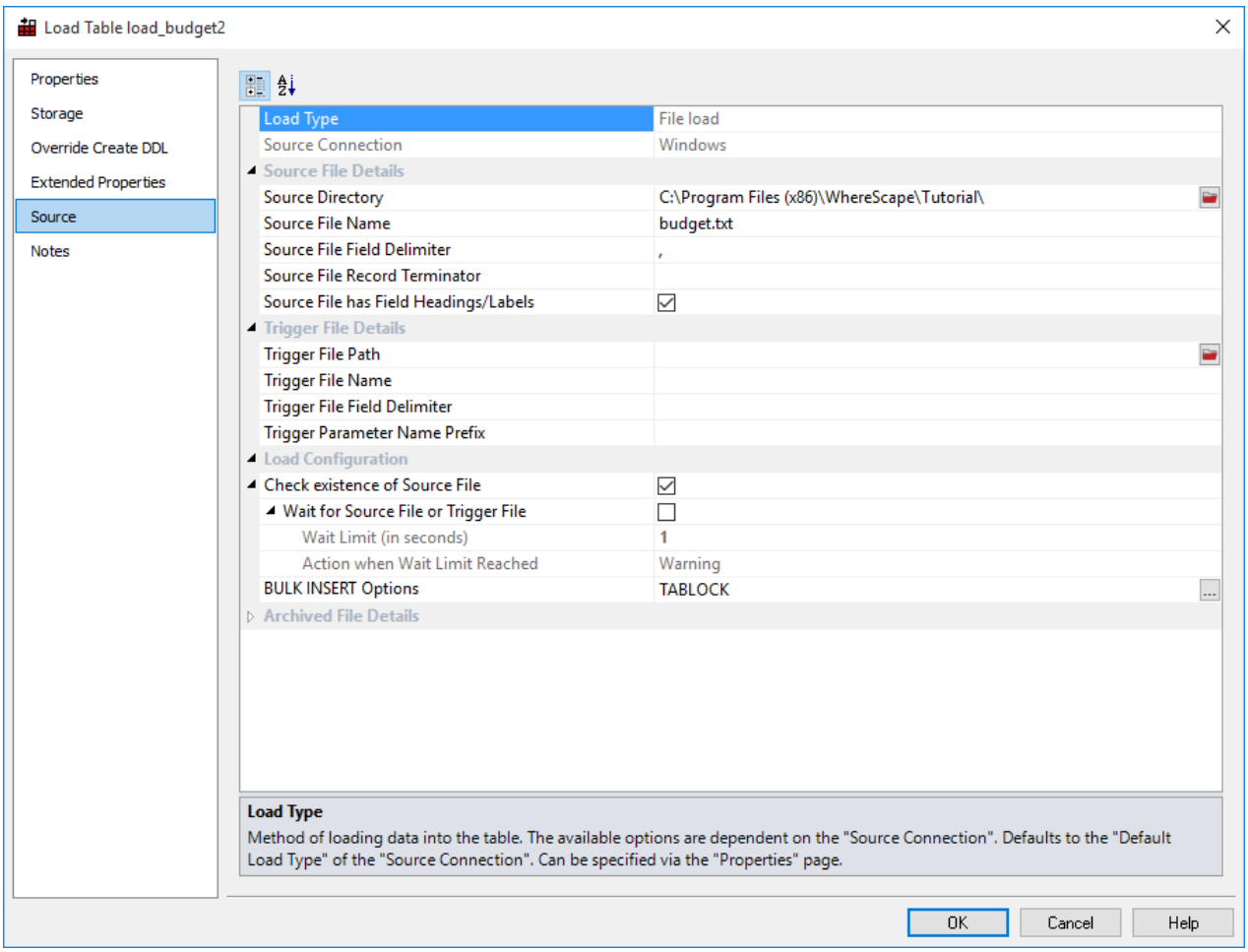
Fields	Description
	separately or can be combined together using one set of enclosing $ such as $YYYYMMDD$.
Archived Trigger File Path	Optional full path (absolute path) of the folder/directory to MOVE the successfully processed Trigger File on the Windows or UNIX/Linux system.
Archived Trigger File Name	Optional new name to RENAME the successfully processed Trigger File to. By default, the original file name is used it is optional to rename it. However, the in-built variable $SEQUENCE$ can be used to include a unique sequence number in the new name. Likewise, the in-built variables $YYYY$, MM, DD, HH, MI and/or SS can be used to include the number of the current year, month, day, hour, minute and/or second respectively. The date/time components can be used separately or can be combined together using one set of enclosing $ such as $YYYYMMDD$.
PDW File Load Options (Configuration details specific to PDW File Loads)	
Load Mode	Select the loading mode from the list. The available options include: Append/Fast Append/Reload/Upsert. Note that selecting the Upsert Load Mode requires a business key to be defined.
Use Staging database	If this field is set and a staging database is defined in the target connection, then the database will be used for staging, otherwise the target connection's database will be used for staging. To set a Staging Database in the target connection, refer to **Connections - Database** for details.
Additional Command Line Parameters	Optional additional command line parameters (space separated) for PDW file loads which are not configured elsewhere. Such additional configurations include: certificate verification (-N), skipping loads of empty files (-se), specifying the batch load size (-b <batchsize>), white space trimming around string fields (-c), converting empty strings to null (-E), and load failure options (-rt value|percentage, -rv <reject_value>, -rs <reject_sample_size>).
SQL Server Integration Services (SSIS)	SQL Server Integration Services (SSIS) Attributes.
SSIS Source-Identifier Case Conversion	Case-sensitivity conversion applied to Source Object Identifiers (such as table, view, and column names) in RED-generated SSIS packages. If no conversion is applied then the exact case of the identifier defined in the RED metadata is used in SSIS.
SSIS Set Source-Code Page	Enables the SSIS source code page property.
SSIS Source Code Page	Use this field to specify the source code page to use when unavailable for the data source.
SSIS Destination-AlwaysUseDefaultCodePage	Forces the use of the Default CodePage property value when describing character data.
SSIS Set Destination-Code Page	Enables the SSIS destination code page property.
SSIS Destination-Code Page	Use this field to specify the destination code page to use when unavailable for the data source.
SSIS Row Count Log	During an SSIS Load, include Row Count logging.
SSIS Acquire Table Lock	Specify whether the destination table is locked during the load. This allows SQL Server to reduce lock escalation overheads during loading and will promote minimal logging in SQL 2008+ when a Bulk Logged recovery model is used for the database. The default value of this property is true.

© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
SSIS Commit Interval	Specify the batch size that the OLE DB destination tries to commit during fast load operations. The default value of 2147483647 indicates that all data is committed in a single batch after all rows have been processed. If you provide a value for this property, the destination commits rows in batches that are the smaller of (a) the Maximum insert commit size, or (b) the remaining rows in the buffer that is currently being processed.
Set SSIS Batch Size	Use this field to specify the number of rows in a batch. The default value of this property is empty, which indicates that no value has been assigned.
SSIS Rows per Batch	Use this field to define the maximum number of rows in a SSIS batch.

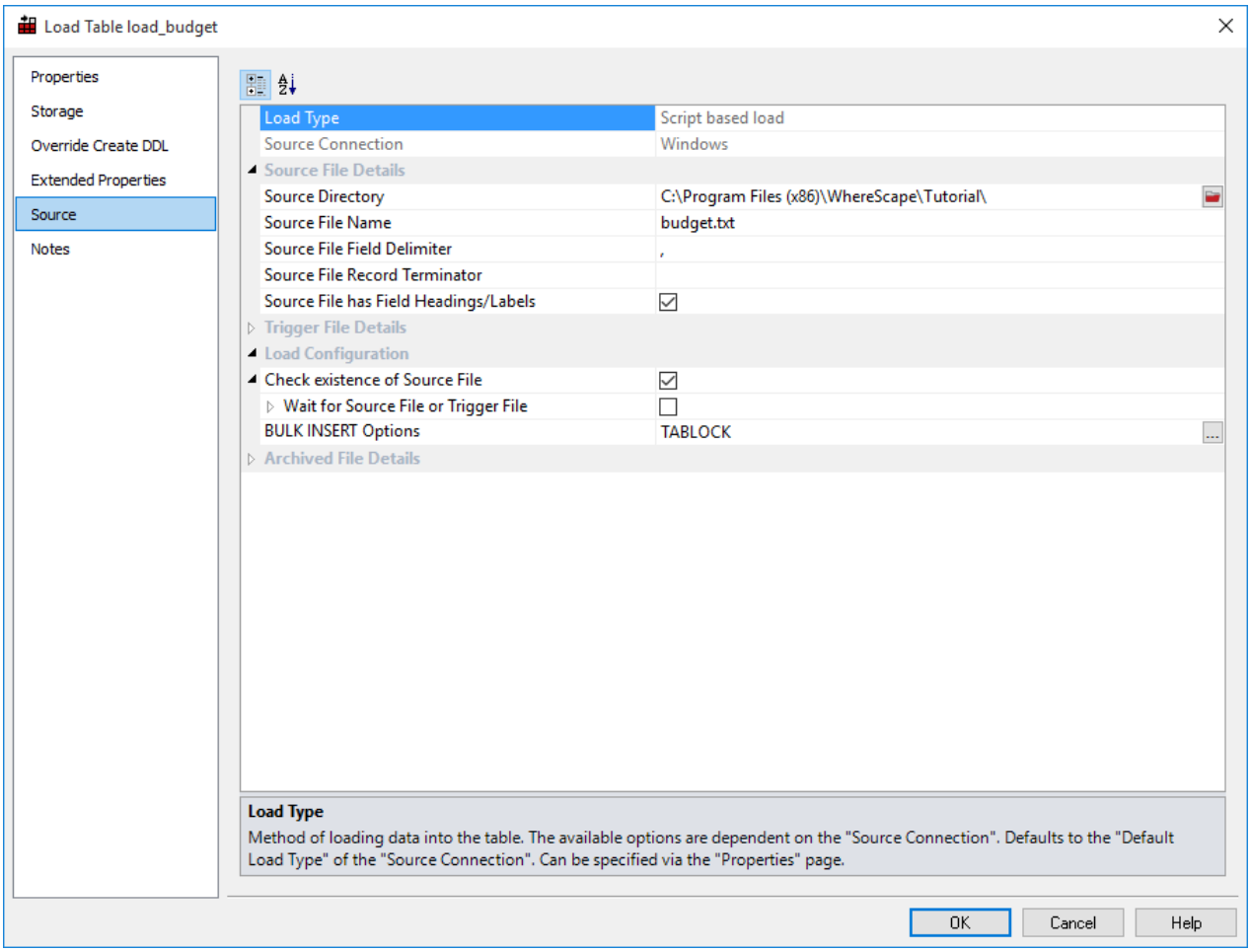
File and Script based load source screens

SQL Server File load Source screen:



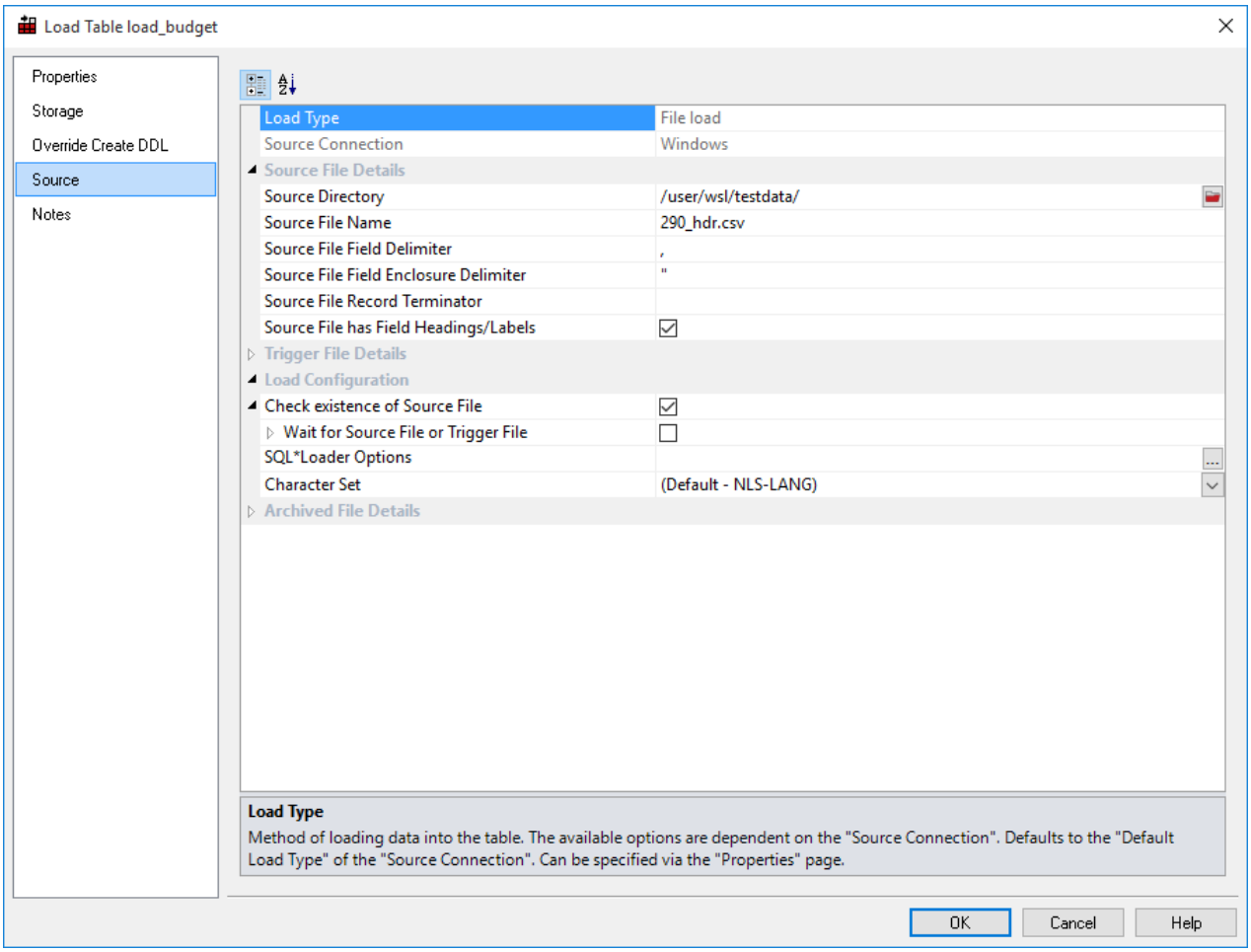
© Copyright, WhereScape Software Ltd All rights reserved.

SQL Server Script load Source screen from a Windows connection:



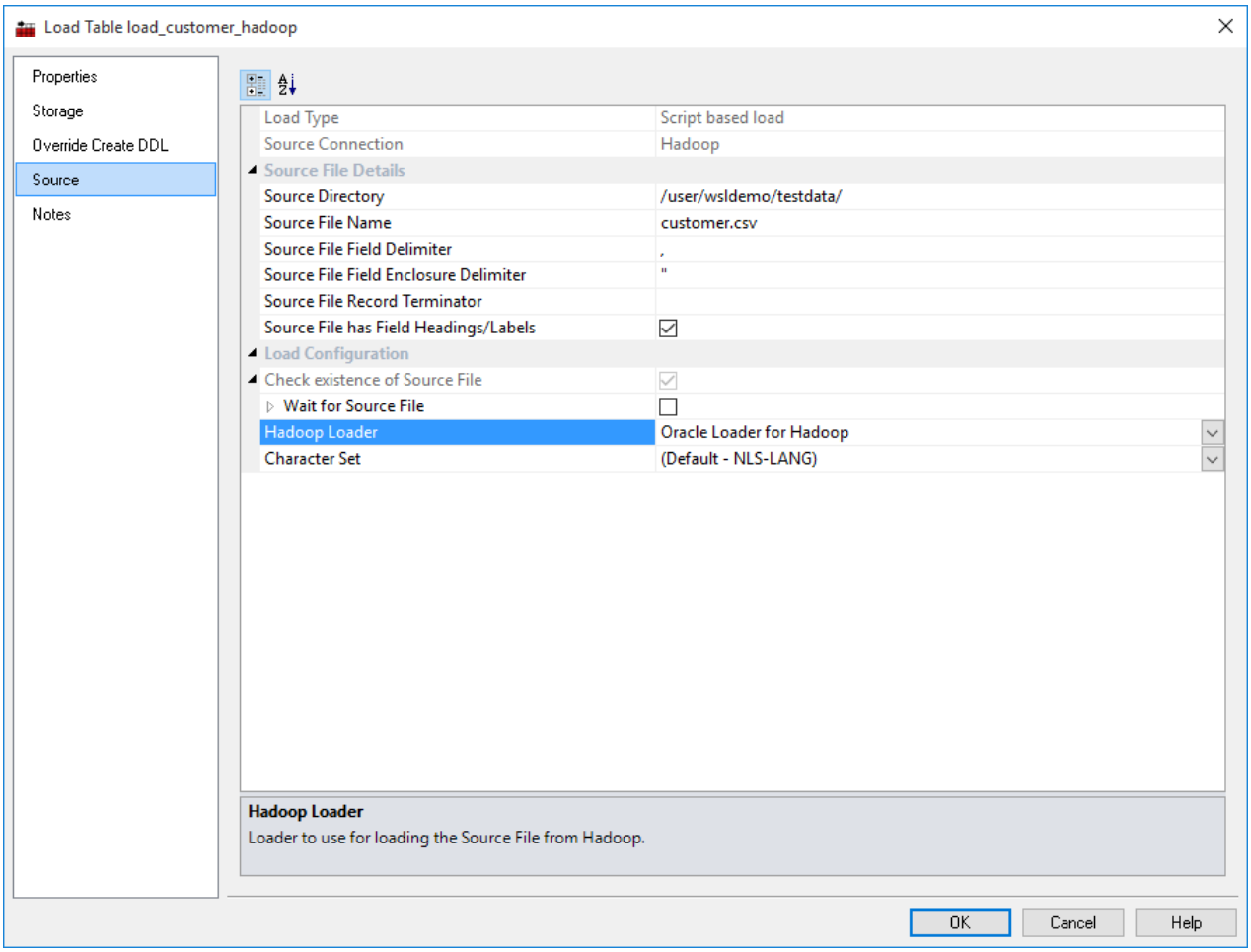
© Copyright, WhereScape Software Ltd All rights reserved.

Oracle File load Source screen:



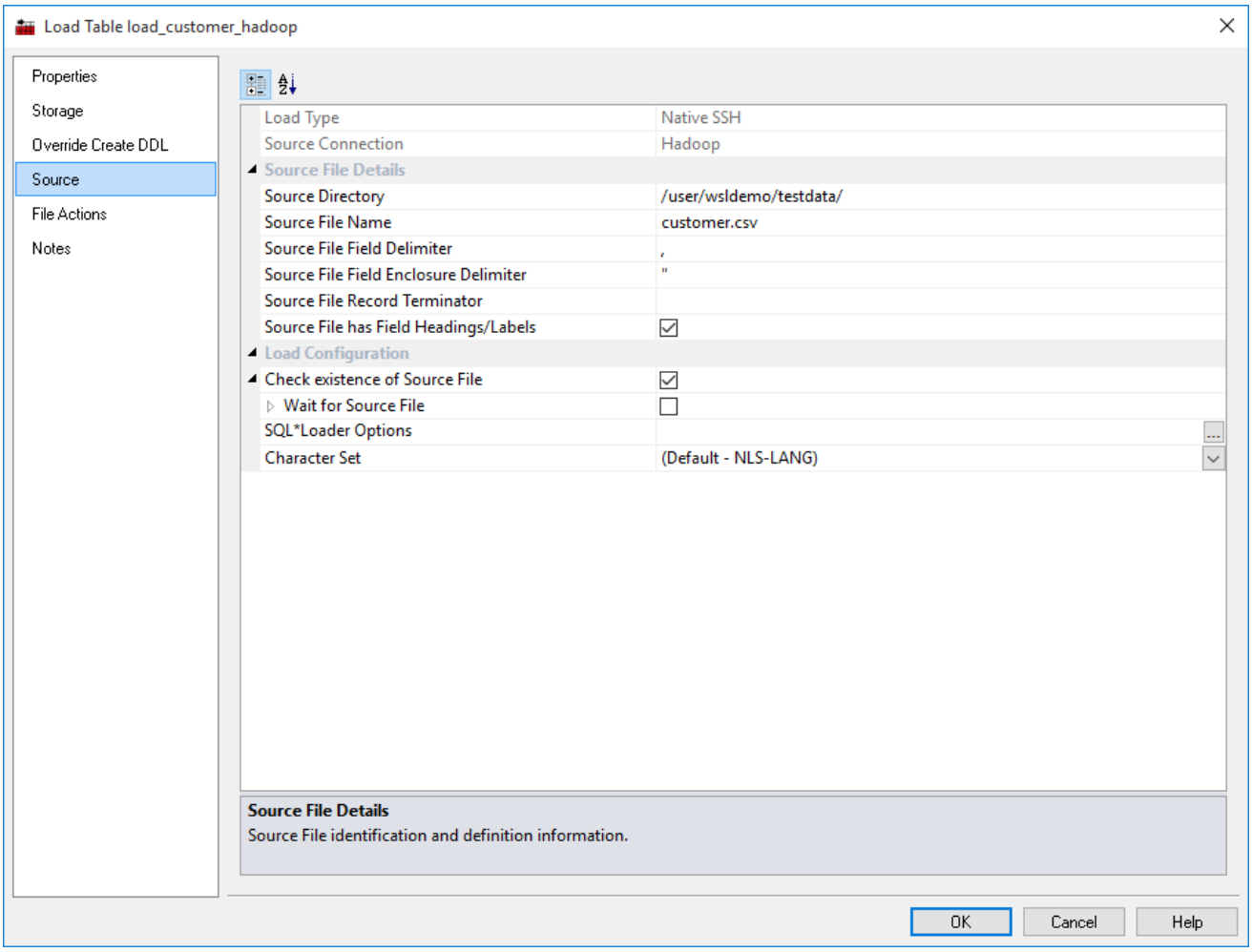
© Copyright, WhereScape Software Ltd All rights reserved.

Oracle Script Load Source screen from a Hadoop connection:



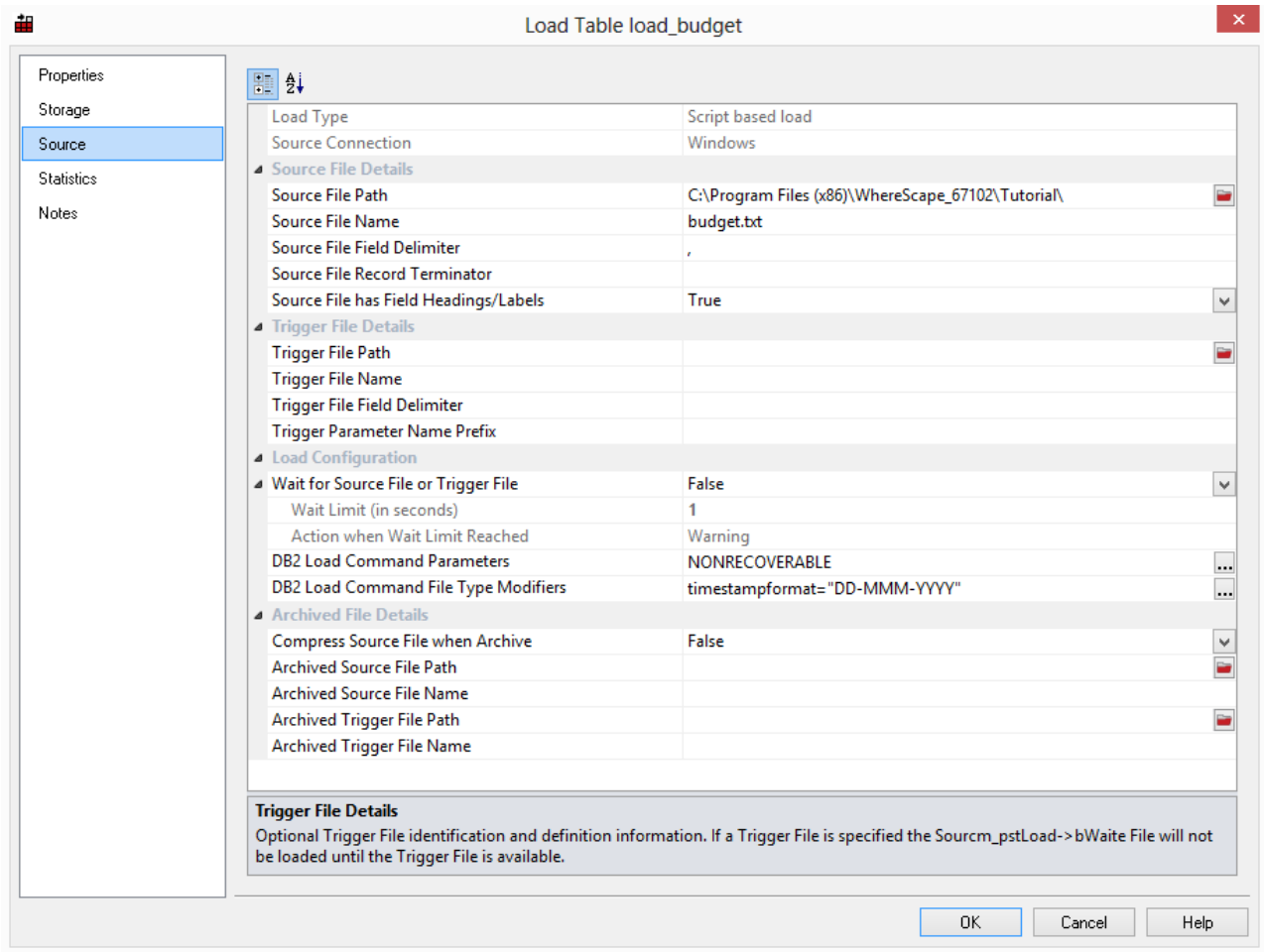
© Copyright, WhereScape Software Ltd All rights reserved.

Oracle Native SSH Load Source screen from a Hadoop connection:



© Copyright, WhereScape Software Ltd All rights reserved.

DB2 File load Source screen:



Script Based Load

A script based load table must have a Host Script defined. During the load process this host script is executed and the results returned.

During the 'drag and drop' creation of a load table from a UNIX or Windows file a script can be generated by selecting one of the 'Script based' load options. This script can then be edited to fully meet any requirements.

Note

If the UNIX Scheduler is not being used and a UNIX script is generated and executed interactively, the following two variables will be setup as part of the environment.

- DSS_USER=username
- DSS_PWD=password

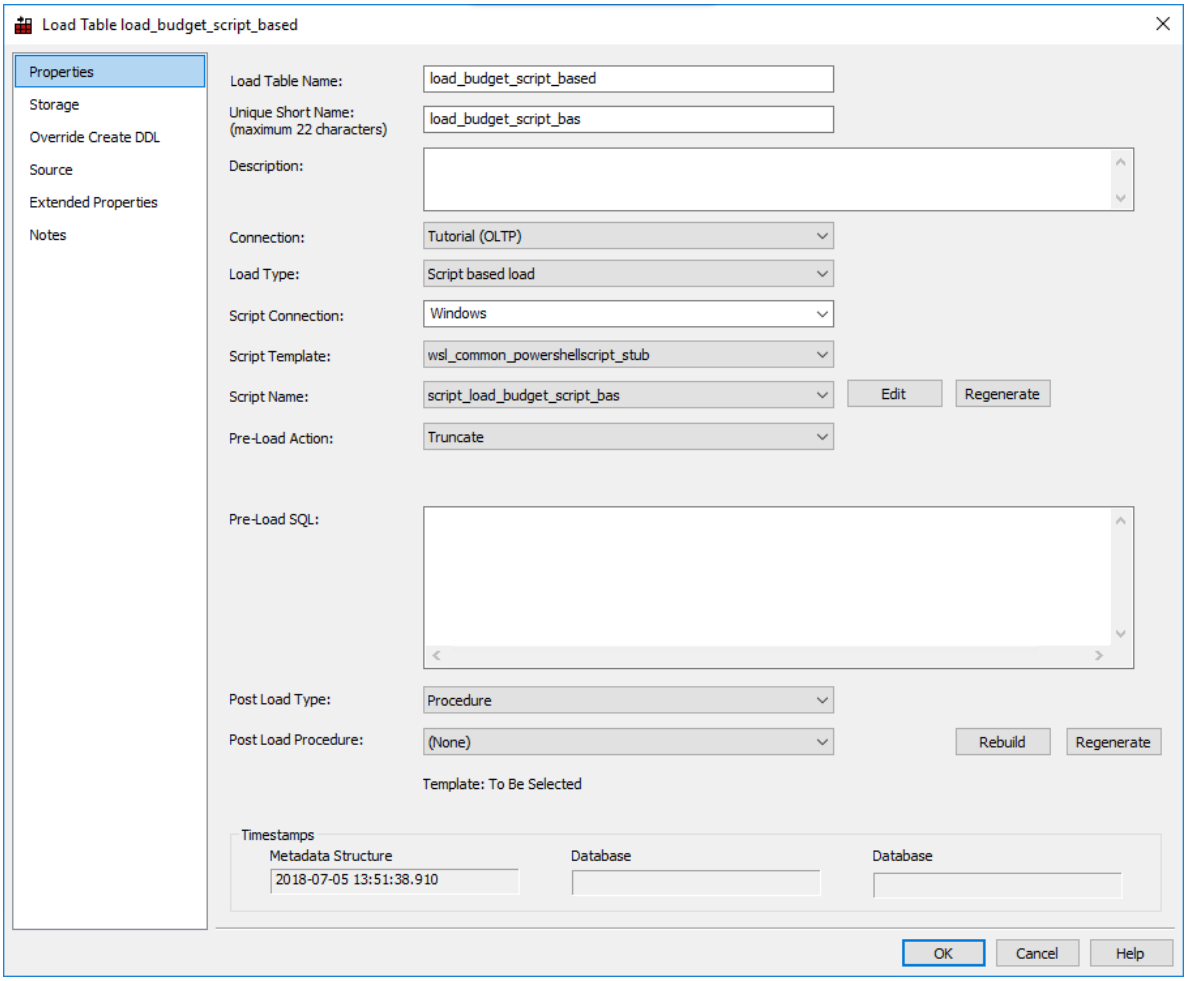
Where 'username' and 'password' are the entries for the data warehouse database.

There are a number of conventions that must be followed if these host scripts are to be used by the WhereScape scheduler.

1. The first line of data in **standard out** must contain the resultant status of the script. Valid values are '1' to indicate success, '-1' to indicate a Warning condition occurred but the result is considered a success, '-2' to indicate a handled Error occurred and subsequent dependent tasks should be held, -3 to indicate an unhandled Failure and that subsequent dependent tasks should be held.
2. The second line of data in **standard out** must contain a resultant message of no more than 256 characters.

© Copyright, WhereScape Software Ltd All rights reserved.

3. Any subsequent lines in **standard out** are considered informational and are recorded in the audit trail. The normal practice is to place a minimum of information in the audit trail. All bulk information should be output to **standard error**
4. Any data output to **standard error** will be written to the error/detail log. Both the audit log and detail log can be viewed from the WhereScape RED tool under the **Scheduler** window.
5. When doing **Script based loads**, it is easy to use the **Regenerate** button beside the **Script Name** field to regenerate the scripts. You can then click the **Edit** button to view and edit the generated script, as per your requirements.



Note

Script-based loads on Windows supports both DOS Batch and **PowerShell scripts**.

File Wild Card Load

The loading of multiple files in one pass is best undertaken as a host script.

A generated host script contains the code to handle wild card loads and the subsequent loading and renaming of multiple files. The process for setting up wild card loads is as follows:

1. Drag one of the files into a **Load Table** target and select one of the **Script** based options.
2. If necessary edit the generated script and replace the load file name with the required wild card version. (For example test_result_234.txt renamed to test_result_*.txt. This load file name is defined as a variable at the start of the script.
3. If renaming or moving to another directory is required, then you may need to uncomment the rename phase and enter the desired directory or rename syntax.

© Copyright, WhereScape Software Ltd All rights reserved.

4. If only one file at a time is to be processed, but still based on a wild card file name, then you may need to uncomment the break statement at the bottom of the script. This will result in the loading of only the first file encountered that matches the wild card criteria.

In every event, check the generated script as it will contain instructions on how to handle wild cards for the appropriate script type (Windows or UNIX).

XML File load

XML file loads are only supported from a Windows connection. There are multiple formats for data exchange when using XML. See the sections below for details on how an XML file is handled.

To load an XML file located in a Windows directory, proceed as follows:

1. Create a connection to the Windows system.
2. Browse to the connection and locate the XML file.
3. Make Load tables the middle pane drop target by double-clicking on the **Load Table** object group in the left pane.
4. Drag the XML file into the middle pane.

The only rules concerning the xml file are that the data element tags are the column names and each row of data is a child of the root element.
For example:

```
<row>
<dim_customer_key>7</dim_customer_key>
<code>228</code>
<name>JOHN AND JOES TOYS</name>
<address>3700 PARNELL RISE</address>
<city>BEAVERTON</city>
<state>OR</state>
<dss_source_system_key>1</dss_source_system_key>
<dss_update_time>2003-10-03T10:02:15.310</dss_update_time>
</row>
```

Supported XML Formats

WhereScape RED supports two types of xml file construct. The normal xml standards have the data in the xml file and the table definitions in a separate xsd (xml schema definition) file which is only required when the table is being created or when the xml file is being validated for form. An alternate standard is used by Microsoft. This second standard is an in line definition which produces one file which contains a Schema element in the data stream where the column names and their approximate data types are defined.

Separate XML and XSD files

The normal XML standards have the data in the xml file and the table definitions in a separate xsd (xml schema definition) file which is only required when the table is being created or when the xml file is being validated for form. The xsd file name is found within the xml file in an xsi (xml schema instance) statement which can include a namespace definition; for example:

```
<root xmlns="http://www.wherescape.com/wsl-schema"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:schemaLocation="http://www.wherescape.com/load_table.xsd">
or no namespace;e.g.
<root xmlns="http://www.wherescape.com/wsl-schema"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
xsi:noNamespaceSchemaLocation="load_table.xsd">
```

The xsd file is an xml file and should be found in the same directory as the xml file that calls it. This xsd file will contain the column definitions for the load table which will be defined during the drag and drop.

The column definitions within the xsd file must be detailed enough to define a load table that the xml file can be loaded into. The data type mapping between the xsd file and the database have been implemented as below:

XSD	SQL Server	Oracle	IBM DB2	Teradata
string with length	char()	char()	char()	char()
string with maxlength	varchar()	varchar2()	varchar()	varchar()
integer	integer	integer	integer	integer
decimal with precision and scale	numeric(x,y)	number(x,y)	decimal(x,y)	numeric(x,y)
dateTime (ISO8601)	datetime	date	timestamp	timestamp
i2	integer	integer	integer	integer
i4	integer	integer	integer	integer
r4	float	number	float	varchar(40)
r8	float	number	float	varchar(40)
float	float	number	float	varchar(40)

These are the ISO-ANSI SQL/XML standards and in the case of integers, datetime and floats the column can be defined with one line. i.e.

```
<xsd:element name="Policy_ID" type="xsd:integer"/>
<xsd:element name="Quote_Date" type="xsd:dateTime"/>
<xsd:element name="Quote_Price" type="xsd:r4"/>
```

In the case of strings and decimals the column requires a bit more detail to produce the correct data type. Strings can be fixed length with padded data by using the length attribute.

The following will produce a char(1) column called Excess_Waiver:

```
<xsd:element name="Excess_Waiver">
<xsd:restriction base="xsd:string">
<xsd:length value="1"/>
</xsd:restriction>
</xsd:element>
```

Strings can be of variable length by using the maxLength attribute. The following produces a column of SQL Server varchar(8) or Oracle varchar2(8) called Password:

```
<xsd:element name="Password">
<xsd:restriction base="xsd:string">
<xsd:maxLength value="8"/>
</xsd:restriction>
</xsd:element>
```

Decimal numbers are defined with the precision and scale attributes. If the scale attribute is zero or missing then the column will be a whole number of size precision. The following produces a SQL Server column of numeric(6) or an Oracle column of number(6):

© Copyright, WhereScape Software Ltd All rights reserved.

```
<xsd:element name="code" >
<xsd:restriction base="xsd:decimal">
<xsd:precision value="6"/>
<xsd:scale value="0"/>
</xsd:restriction>
</xsd:element>
```

The following produces a SQL Server column of numeric(8,2) or an Oracle column of number(8,2):

```
<xsd:element name="code" >
<xsd:restriction base="xsd:decimal">
<xsd:precision value="8"/>
<xsd:scale value="2"/>
</xsd:restriction>
</xsd:element>
```

An example file with most data types would be as follows:

```
<xsd:schema xmlns="http://www.wherescape.com/wsl-schema"
      xmlns:xsd="http://www.wherescape.com/XMLSchema">
<xsd:element name="Col_name1" type="xsd:integer"/>
<xsd:element name="Col_name4" type="xsd:dateTime"/>
<xsd:element name="Col_name5">
<xsd:restriction base="xsd:string">
<xsd:maxLength value="100"/>
</xsd:restriction>
</xsd:element>
<xsd:element name="Col_name6">
<xsd:restriction base="xsd:string">
<xsd:length value="100"/>
</xsd:restriction>
</xsd:element>
<xsd:element name="Col_name7" type="xsd:float"/>
<xsd:element name="Col_name8" >
<xsd:restriction base="xsd:decimal">
<xsd:precision value="6"/>
<xsd:scale value="2"/>
</xsd:restriction>
</xsd:element>
</xsd:schema>
```

The column order will be the same as the xsd file.
Any columns which are missing from the row will be NULL in the loaded row.

The dateTime format in the xml file is defined as ISO8601 which looks like this:
2003-10-03T10:02:15.310

- WhereScape RED will load this string into **Oracle** as:
  `TO_DATE('20031003100215','YYYYMMDDHH24MI')`
- Or for **SQL Server**:
  `CONVERT(DATETIME,'2003-10-03T10:02:15.310',126)`

© Copyright, WhereScape Software Ltd All rights reserved.

- Or for **DB2**:
  `TIMESTAMP_FORMAT('20031003100215', 'YYYYMMDDHH24MISS')`
- Or for **Teradata**:
  `TIMESTAMP_FORMAT('20031003100215', 'YYYYMMDDHH24MISS')`

- The xsd file is only required to create the load table, if the load table is only being loaded then this file is ignored.
- To check that the xml and xsd files are well formed you can open them with any web browser. If the files display with no errors then they are valid xml files.

In line schema definition

The other supported xml construct allows the use of in line schema definitions as produced by the Microsoft FOR XML AUTO, ELEMENTS, XMLDATA query. This will produce one file which contains a Schema element in which the column names and their approximate data types are defined. Because the supplied data types are not concise enough to define the table columns correctly, this method will produce load tables of either data type text for SQL Server or varchar2(4000) for Oracle. The column names are taken from the <element type="col_name"/> strings within the Schema element. The data elements will be the same as above with the column names making up the start and end tags and the rows being the children of the root element. The file that is produced by the FOR XML query above needs to be changed slightly to comply with the xml standard. Remove everything before the Schema element and then give the file a starting root element and a closing root element.eg <root> and </root>

The xml files can optionally start with an xml pre process statement. For example:

```
<?xml version="1.0"?>
```

They may also contain xml comments. For example:

```
<!– comments    -->
```

External Load

For an externally loaded table, the only property that is executed is the Post Load procedure.

Any **After** transformations recorded against any of the columns in an Externally loaded table will also be processed.

Post load procedures can either be manually generated from a RED provided procedure outline or generated leveraging a RED template—refer to **Rebuilding Update Procedures** for details.

Note

For custom database targets, the option to use Post Load Scripts is provided, as an alternative to post load procedure. Post load scripts can be generated using a RED template—refer to **Rebuilding Update Procedures** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

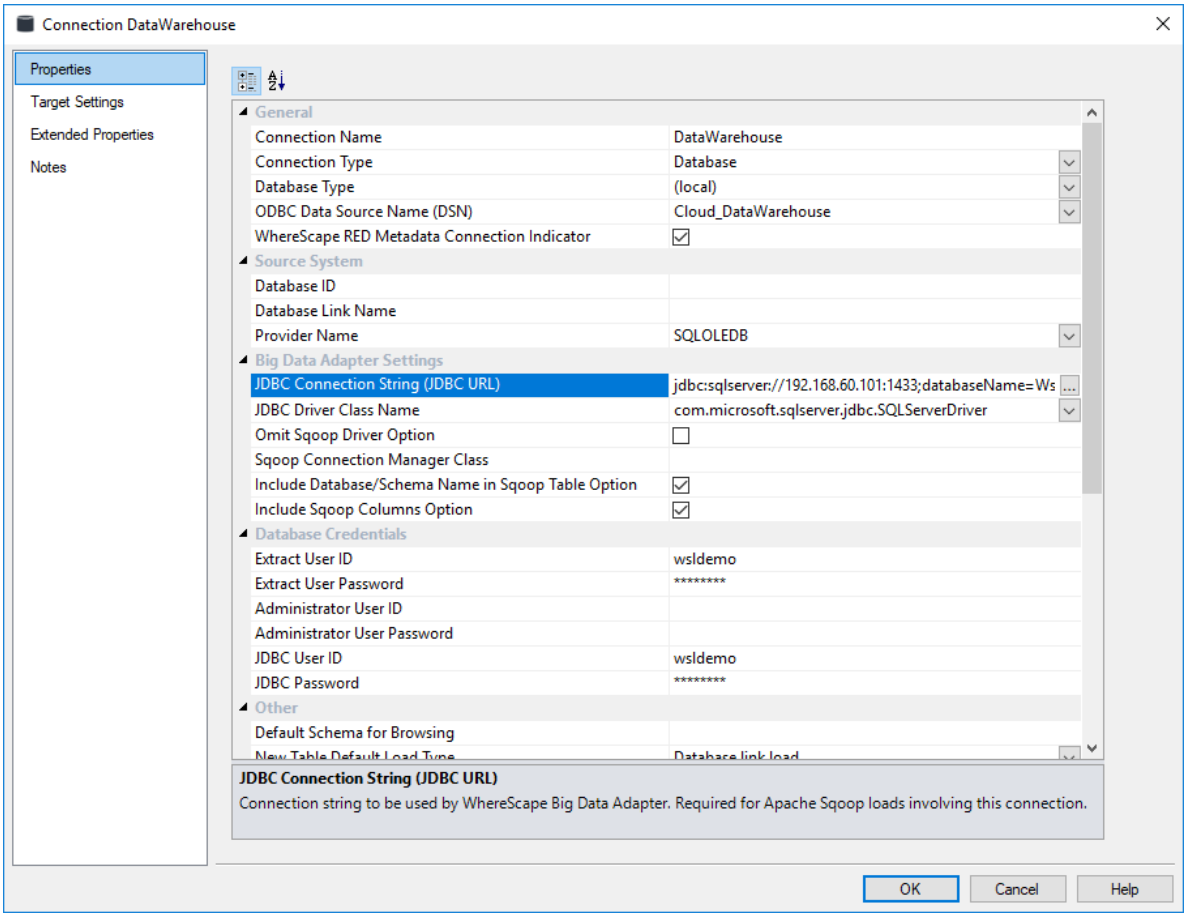
Apache Sqoop Load

The **Apache Sqoop load** type enables loading data directly from Hive/HDFS to any (non-Hive) targets. This load method provides a non database specific method of loading data into any database target, which is especially helpful when loading data directly from HDFS or Hive to SQL Server and Oracle target databases.

Apache Sqoop load from a Hive connection to a Hive target are not supported.

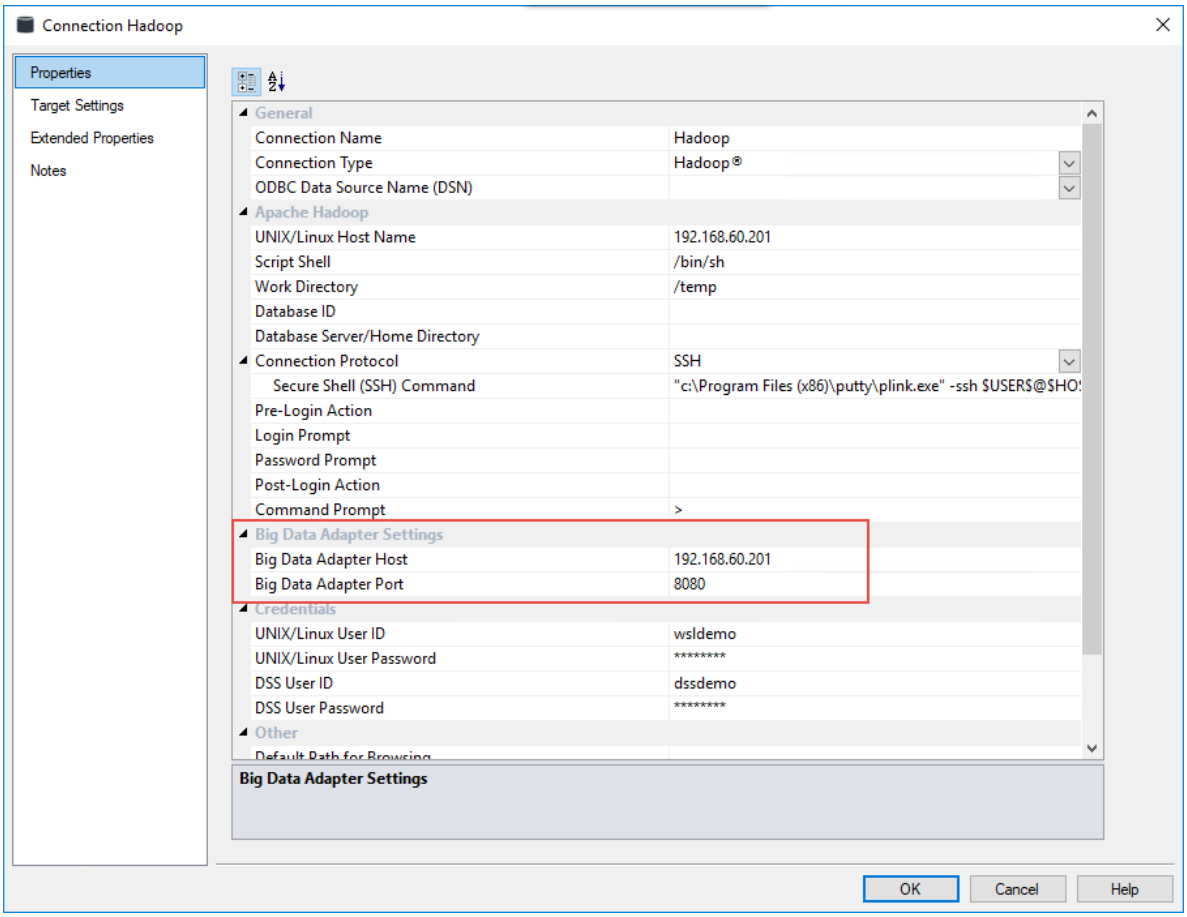
To load tables directly from HDFS/Hive into SQL Server and Oracle target databases:

1. Ensure the relevant **Data Warehouse connection** has the following fields set:
   - JDBC Connection string (JDBC URL)
   - JDBC Driver Class Name
   - Omit Sqoop Driver Option - tick this check-box for loads into Oracle target databases
   - JDBC User ID
   - JDBC Password

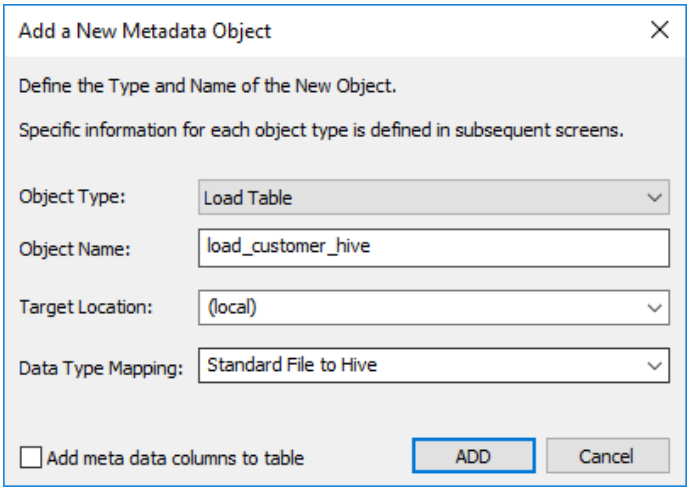


2. When loading from **Hadoop connections**, please ensure the Hadoop connection has its **BDA server host** and **port** fields set, in addition to Hive connections.

© Copyright, WhereScape Software Ltd All rights reserved.



3. Browse the desired **Hadoop/Hive connection**.
4. **Drag and Drop** the table from the Hadoop/Hive connection displayed in the right pane into the middle pane.
    - Change the table name if necessary, and select the relevant target location to place the table from the drop-down list.

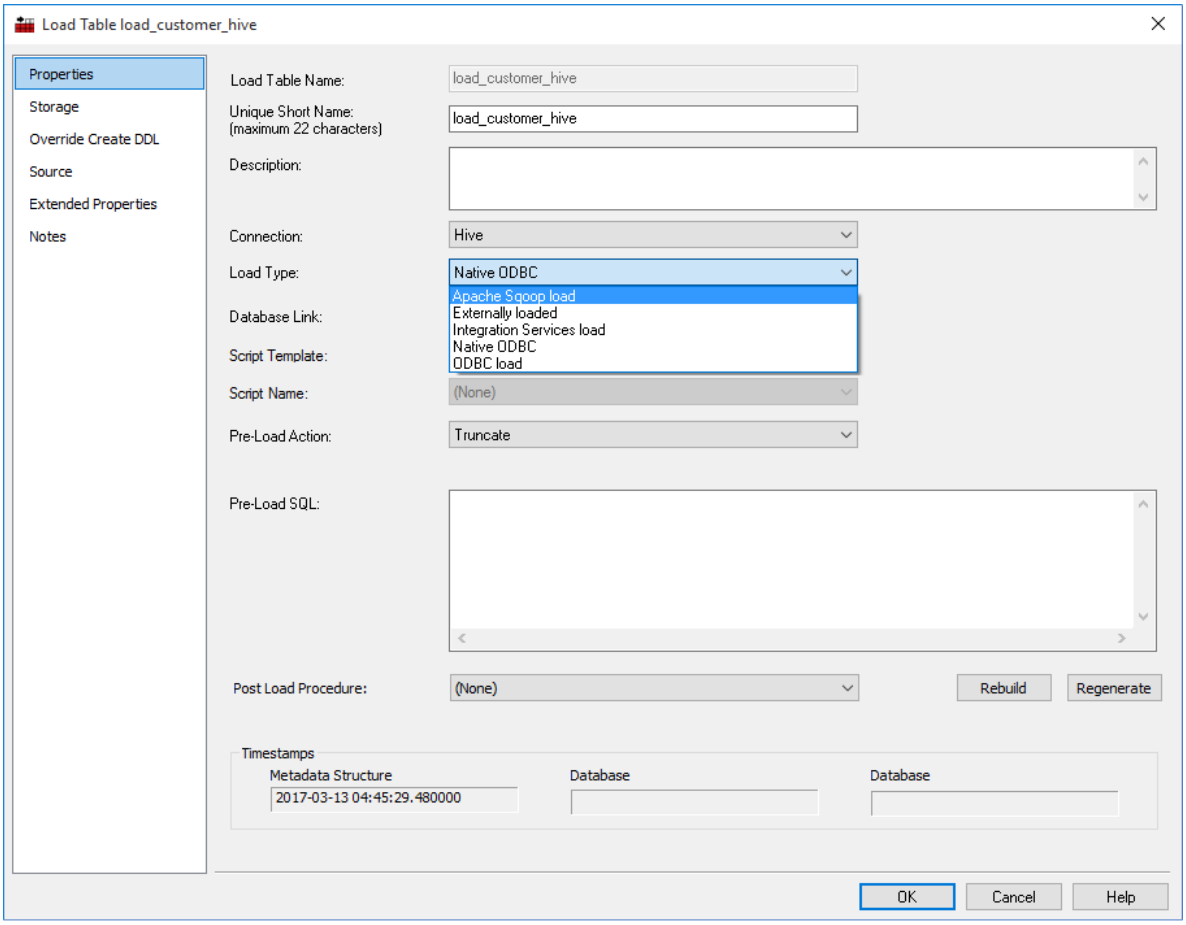


Note:

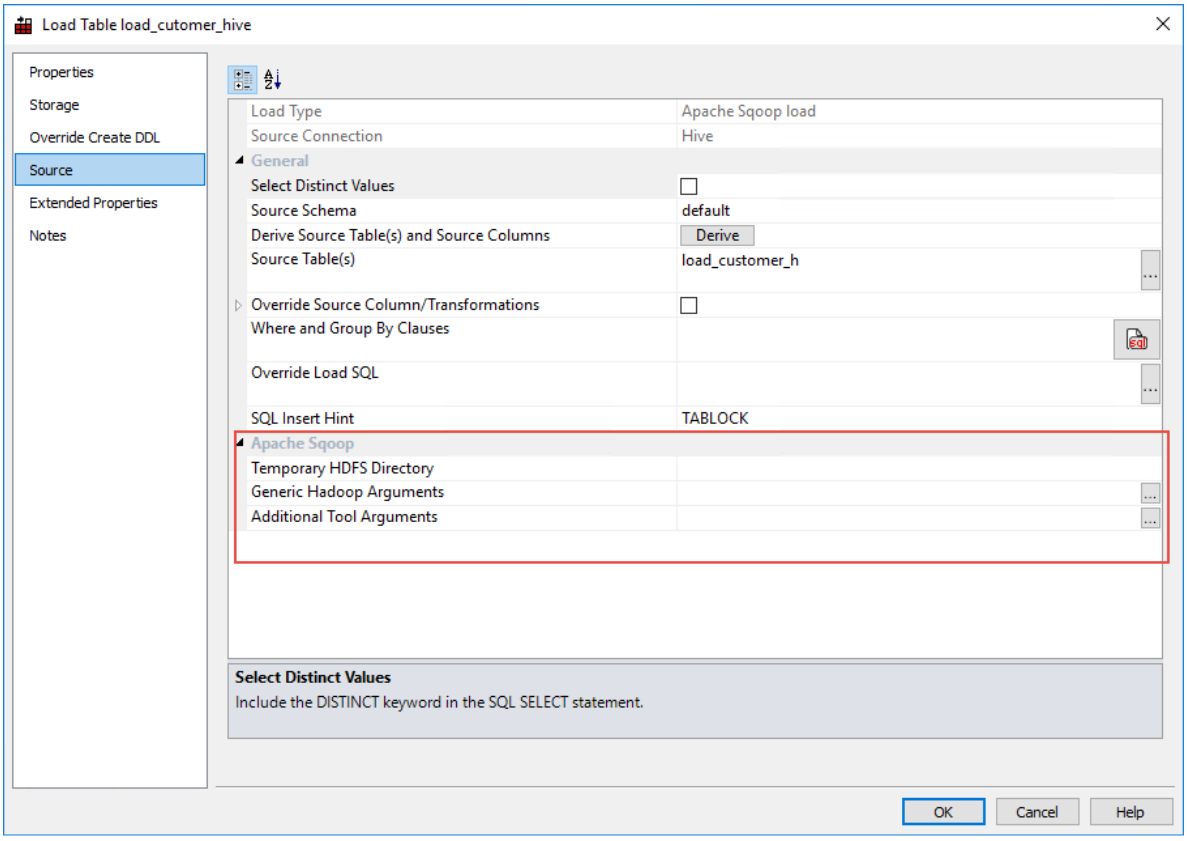
The option **Add meta data columns to table** is used for creating Load tables that are used in creating Data Vault objects. If this option is selected, two DSS columns (**dss_record_source** and **dss_load_date**) are included in the meta data for the table and are populated by transformations. These two DSS columns could equally be applied to other Load tables not used in a Data Vault system, but are particularly important to comply with the Data Vault standards. Refer to **Data Vaults** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

5. Select **Apache Hadoop Load** from the **Load Type** drop-down list.



6. Click the **Source** tab to add any Apache Sqoop specific options:



© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
Temporary HDFS Directory	Loading from a Hive source to the data warehouse is implemented in two steps. First, the data is extracted from the Hive table into a temporary HDFS directory. Then, this temporary directory is loaded using "sqoop export". The location of the temporary directory can be configured in RED on the source tab of the load table. When this field is left blank, the default is "/tmp".
Generic Hadoop Arguments	This field enables you to add additional arguments just after the Sqoop command keyword, in this case it is import, in the Sqoop command line.
Additional Tool Arguments	This field enables you to add additional arguments after the generated Sqoop command line.

7. Click **Create and Load** to create and load the table.

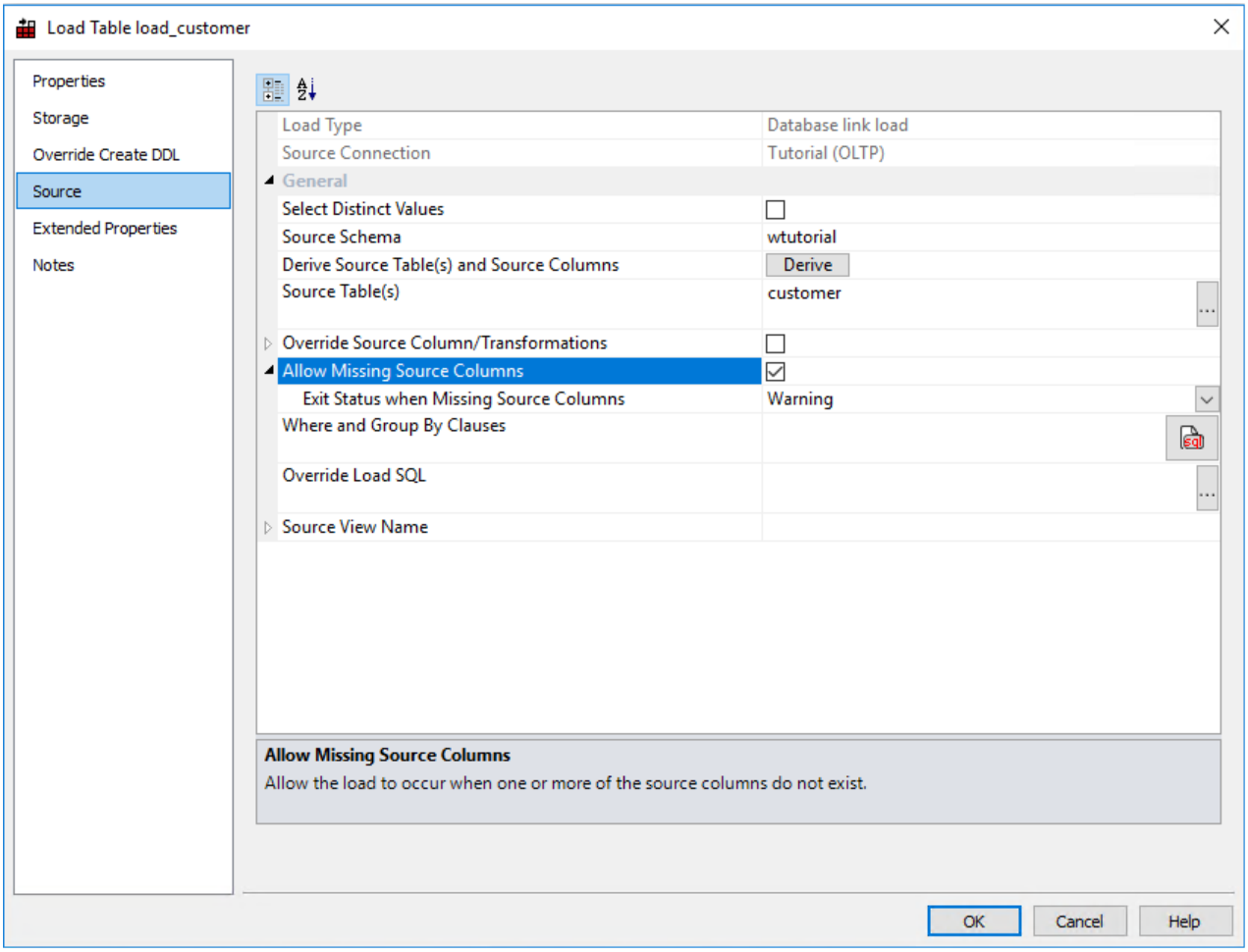
Handling Missing Source Columns

Note

This option is only available when running an **Oracle** data warehouse.

By default, a load fails if a source column that is to be loaded does not exist. This default action can be modified by using the **Allow Missing Source Columns** feature of a load table.
When this feature is enabled the load process will check all source columns, and if any are found to be missing will replace those missing columns with a Null value (which can be changed, see below).

On the **Source** tab of a Load table's Properties there are two check boxes and a drop-down list that we will cover in this section. See the following example:



© Copyright, WhereScape Software Ltd All rights reserved.

<table>
<tr><th>Fields</th><th>Description</th></tr>
<tr><td>Override Source Column/Transformation</td><td>

When this is enabled, the Source Columns edit window is enabled. With this enabled, a load table uses the contents of the Source columns to specify which columns are being extracted during a table load.
When this is unselected the load process builds up the statement to select from the source table(s) by looking at the two fields source table and source column and any transformations associated with each column in the load table. If a transformation is present then the transformation overrides the contents of source table/source column. See the following section for more information on transformations.

Note
This box must be unselected to enable Allow Missing Source Columns support.
</td></tr>
<tr><td>Allow Missing Source Columns</td><td>
When this is enabled, the load process will examine every column in the load table. Based on the source table/source column fields associated with each column, it will check to see if the column exists in the source database. If the column exists normal processing will occur. If the column does not exist then a Null will be substituted for the column, and the load will proceed.

If one or more columns are found to be missing the load process reports this situation. The status level of this reporting can be set via the Exit status drop-down. See the following topic. In all cases the load will be deemed to have been successful provided that no other errors occur.

Often Null values are not desirable in a data warehouse. This Null value can be replaced by some other value by means of a During or After transformation. For example, a During transformation, as shown below, set on a missing column called 'State' will replace the Null with the value 'N/A'.

NVL(state,'N/A')
</td></tr>
<tr><td>Exit Status When Missing Columns</td><td>
If columns are found to be missing as a result of a Non Mandatory check then a message is recorded in the Audit trail and against the task if running in the Scheduler. The drop-down list of the same name allows the choice of action in the event of a missing column or columns. The choices are:
<table>
<tr><th>Choice</th><th>Impact</th></tr>
<tr><td>Success</td><td>The message informing that columns are missing uses an information status.</td></tr>
<tr><td>Warning</td><td>The message issued is a warning. The table load will be identified as successful but with warnings.</td></tr>
<tr><td>Error</td><td>The message issued is an error. The table load will still complete successfully albeit with an error message.</td></tr>
</table>
This drop-down list is only available if the Non Mandatory Source Columns check-box is ticked.
</td></tr>
<tr><td>Limitations of Missing Column check</td><td>
The check for missing columns will only examine and check those columns that are recorded in the source table and source column fields of a load table's column Properties. Therefore if a source column is used in a transformation or join, but is not recorded in the two fields mentioned above it will not be checked.

If a column is used in a transformation, it should have the same usage (name and case) as is found in the source column and source table fields.

This check has no upper limit on the number of missing columns. All columns can be missing and the load will still succeed.
</td></tr>
</table>

© Copyright, WhereScape Software Ltd All rights reserved.

Load Table Transformations

Each load table column can have a transformation associated with it.

Refer to **Transformations** and **Load Table Column Transformations** for details.

Post-Load Procedures

If a procedure name is entered in the **Post Load Procedure** drop-down field of a Load table's Properties, then this procedure will be executed after the load has completed and after any **after transformations** have occurred.

Click the **Rebuild** button to start the process of generating a new procedure. The **Regenerate** button is used to regenerate the Load table's existing post load procedure.

Refer to **Load Table Column Transformations** for details.

Post load procedures can either be manually generated from a RED provided procedure outline or generated leveraging a RED template—refer to **Rebuilding Update Procedures** for details.

Note

For custom database targets, the option to use Post Load Scripts is provided, as an alternative to post load procedure. Post load scripts can be generated using a RED template—refer to **Rebuilding Update Procedures** for details

Changing Load Connection and Schema

The connection associated with a load table can be changed through the Properties of that table.

Connections can also be changed in bulk by using the following process:

1. Double-click the **Load** table object group in the left pane. This displays all the Load tables in the middle pane.
2. Select the Load tables that you want to change, using standard Windows selection.
3. Right-click to bring up the context menu and select **Change Connect/Schema**.
4. Select the new connection to change all the selected Load tables.

Note

You cannot change the connection **type** but it is possible to change from Database to ODBC connections when the following considerations are taken into account.

Switching Connection from ODBC to Database

This switch should be successful in most cases, although it may not provide the most performant load possible. By default ODBC connections use the source table/column transformation loading method as dates and potentially other data types need to be converted.

When switching to a database link load any transformations will still occur although they may no longer be necessary.

Switching Connection from Database to ODBC

Although it is possible to switch from a Database connection to an ODBC connection, the resultant extract may not function correctly. The main issue is around the conversion of date fields under Oracle. SQL Server tends to handle the date conversion better in most cases. When an extract occurs over an ODBC connection the date is converted in the extract from the source system into a standard ASCII format. This ASCII formatted date is then converted back when it is loaded into the load table. To resolve this problem place a transformation on any date fields. Examples of typical Oracle transformations are:

to_date('"||to_char(ORDER_DATE,'YYYYMMDD')||'","YYYYMMDD")

to_char(ORDER_DATE,'YYYY-MM-DD HH24:MI:SS')

© Copyright, WhereScape Software Ltd All rights reserved.

where ORDER_DATE is the name of the column being extracted. In the first example the conversion string is enclosed within quote marks (') so that it is passed as a literal to the select statement that will be built. Quote marks within the conversion string are protected from premature evaluation by the inclusion of a second quote mark alongside the original.

Note

If a load table is created via drag and drop from an ODBC based connection WhereScape RED will build all the required date transformations.

There are several supplied APIs for changing schema and connections programmatically. Refer to **Callable Routines** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

Dimensions

Dimensions Overview

A dimension table is normally defined, for our purposes, as a table that enables us to constrain queries on the fact table.
A dimension is built from the Data Warehouse connection. Unless you are doing a retro-fit of an existing system, dimensions are typically built from one or more load tables.

The normal steps for creating a dimension are defined below and are covered in this chapter. The steps are:

- Identify the source transactional data that will constitute the dimension. If the data is sourced from multiple tables, ascertain if a join between the source tables is possible, or if a series of lookups would be a better option.
- Using the 'drag and drop' functionality, drag the load table that is the primary source of information for the dimension into a dimension target. Refer to **Building a Dimension** for details.
- If only one table is being sourced and most of the columns are to be used (or if prototyping) you can select the auto create option to build and load the dimension and skip the next 4 steps. Refer to **Building a Dimension** for details.
- Add columns from other load tables if required. Refer to **Building a Dimension** for details.
- Create the dimension table in the database. Refer to **Building a Dimension** for details.
- Build the update procedure. Refer to **Generating the Dimension Update Procedure** for details.
- Run the update procedure and analyze the results. Refer to **Dimension Initial Build** for details.
- Modify the update procedure as required. Refer to **Dimension Initial Build** for details.

Dimension Keys

Dimensions have two types of keys that we will refer to frequently. These are the **Business Key** and the **Artificial Key**.
A definition of these two key types follows:

Business Key

The business key is the column or columns that uniquely identify a record within the dimension. Where the dimension maps back to a single or a main table in the source system, it is usually possible to ascertain the business key by looking at the unique keys for that source table. Some people refer to the business key as the **natural key**. Examples of business keys are:

- The product SKU in a product dimension
- The customer code in a customer dimension
- The calendar date in a date dimension
- The 24 hour time in a time dimension (i.e. HHMM) (e.g.1710)
- The airport short code in an airport dimension.

It is assumed that business keys will never be NULL. If a null value is possible in a business key then the generated code will need to be modified to handle the null value by assigning some default value.

For example the 'Where' clause in a dimension update may become:

SQL Server: Where isnull(business_key,'N/A') = isnull(v_LoadRec.business_key,'N/A')

Oracle: Where nvl(business_key,'N/A') = nvl(v_LoadRec.business_key,'N/A')

DB2: Where coalesce(business_key,'N/A') = coalesce(v_LoadRec.business_key,'N/A')

Note

Business keys are assumed to never be Null. If they could be null it is best to transform them to some value prior to dimension or stage table update. If this is not done, an unmodified update will probably fail with a duplicate key error on the business key index.

© Copyright, WhereScape Software Ltd All rights reserved.

Artificial Key

The artificial key is the unique identifier that is used to join a dimension record to a Fact table. When joining dimensions to Fact tables it would be possible to perform the join using the business key. For Fact tables with a large number of records this however would result in slow query times and very large indexes. As query time is one of our key drivers in data warehouse implementations, the best answer is to always use some form of artificial key. A price is paid in the additional processing required to build the Fact table rows, but this is offset by the reduced query times and index sizes. We can also make use of database specific features such as bitmap indexes in Oracle.

The artificial key is an integer and is built sequentially from 1 upwards.

Refer to **Dimension Artificial Keys** for a more detailed explanation. An artificial key is sometimes referred to as a **surrogate key**.

Building a Dimension

Dimensions are often sourced from one table in the base application. In many cases, there are also codes that require description lookups to complete the de-normalization of the dimensional data.

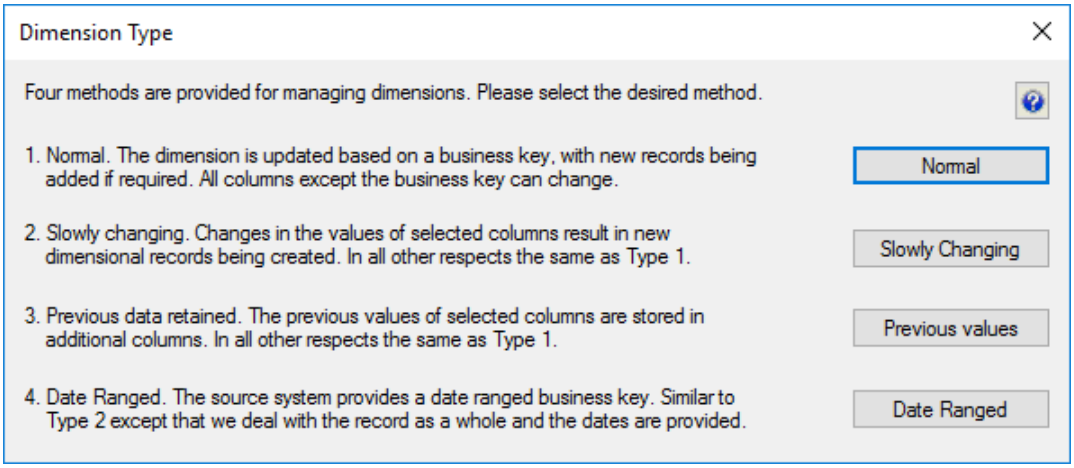
The process for building a dimension is the same for most other tables and begins with the drag and drop of the Load table that contains the bulk of the dimensional information.

Drag and Drop

1. Create a dimension target by double-clicking on the **Dimension group** in the left pane.
2. The middle pane displays a list of all existing dimensions. When this list is displayed in the middle pane, the pane is identified as a target for new dimension tables.
3. Browse to the Data Warehouse via the **Browse > Data Warehouse** menu option.
4. Drag the Load table, that contains the bulk of the dimensional columns, into the middle pane.
5. Drop the table anywhere in the pane.
6. The new object window appears and classifies the new object as a Dimension and provides a default name based on the Load table name. Either accept this name or enter another name and then click **OK** to proceed.

Dimension Type

A dialog appears as shown below. There are four choices for the default generation of the Dimension table and its update procedure.



- The first choice being a **Normal** dimension where a dimensional record is updated and changed whenever any of the non business key information changes.
- The second choice is a **Slowly Changing** dimension where new dimension records are created when certain identified columns in the dimension change.

© Copyright, WhereScape Software Ltd All rights reserved.

Note

For custom database targets, you can set the initial default values for the DSS columns that are used in the procedure generation for slowly changing dimensions. Refer to **DSS Columns for Custom Targets** for details.

- The third choice is a **Previous values** dimension, which enables the storing of the last values of selected fields in secondary columns.
- The fourth choice is a **Date Ranged** dimension, which supports source systems that provide start and end dates.

With any dimension, we identify a **Business Key** that uniquely identifies the dimension records.

For example in the case of the product dimension, the product **code** is deemed to be the business key. The code uniquely identifies each product within the dimension. The product may also have a name or description and various other attributes that distinguish it (e.g. size, shape, color, etc.).

A common question when handling dimensions is what to do when the name or description changes:

- Do we want to track our fact table records based only on the product code? or
- Do we also want to track records based on different descriptions?

An example :

code	description	product_group	sub_group
1235	15oz can of brussel sprouts	canned goods	sprouts

This product has been sold for many years and we consequently have a very good history of sales and the performance of the product in the market. The company does a '20% extra for free' promotion for 3 months during which time it increases the size of the can to 18oz. The description is also changed to be '15 + 3oz can of brussel sprouts'. At the end of the promotion, the product is reverted to its original size and the description changed back to its original name.

The question is, do we want to track the sales of the product when it had a different description (slowly changing) , or should the description of the product simply change to reflect its current name (normal). For this scenario a previous value dimension would not provide much advantage, so it is not discussed.

The decision is not a simple one and the advantages and disadvantages of each of the two choices is discussed below.

Slowly Changing

- Allows the most comprehensive analysis capabilities when just using the product dimension.
- Complicates the analysis. Does not allow a continuous analysis of the product called '15oz can of brussel sprouts' when the description is used. This analysis is however still available through the code which has not changed.
- Adds considerable additional processing requirements to the building of the fact tables that utilize this dimension.
- May track data quality improvements rather than real business change.

Normal

- Does not allow specific analysis of the product during its size change. Note, however that this analysis will probably be available through the combination of a 'promotion' dimension.
- Provides a continuous analysis history for the product called '15oz can of brussel sprouts'. An analysis via description and code will produce the same results.
- Simplifies analysis from an end user's perspective.

As mentioned above, the choice is never a simple one. Even among experienced data warehouse practitioners, there will be a variety of opinions. The decision must be based on the business requirements. In many cases keeping the analysis simple is the best choice, at least in the early stages of a data warehouse development.
Slowly changing dimensions do have a place, but there is nearly always an alternate method that provides equal or better results. In the example above, a promotion dimension coupled with the product dimension

© Copyright, WhereScape Software Ltd All rights reserved.

could provide the same analysis results whilst still keeping product only analysis simple and easy to understand.

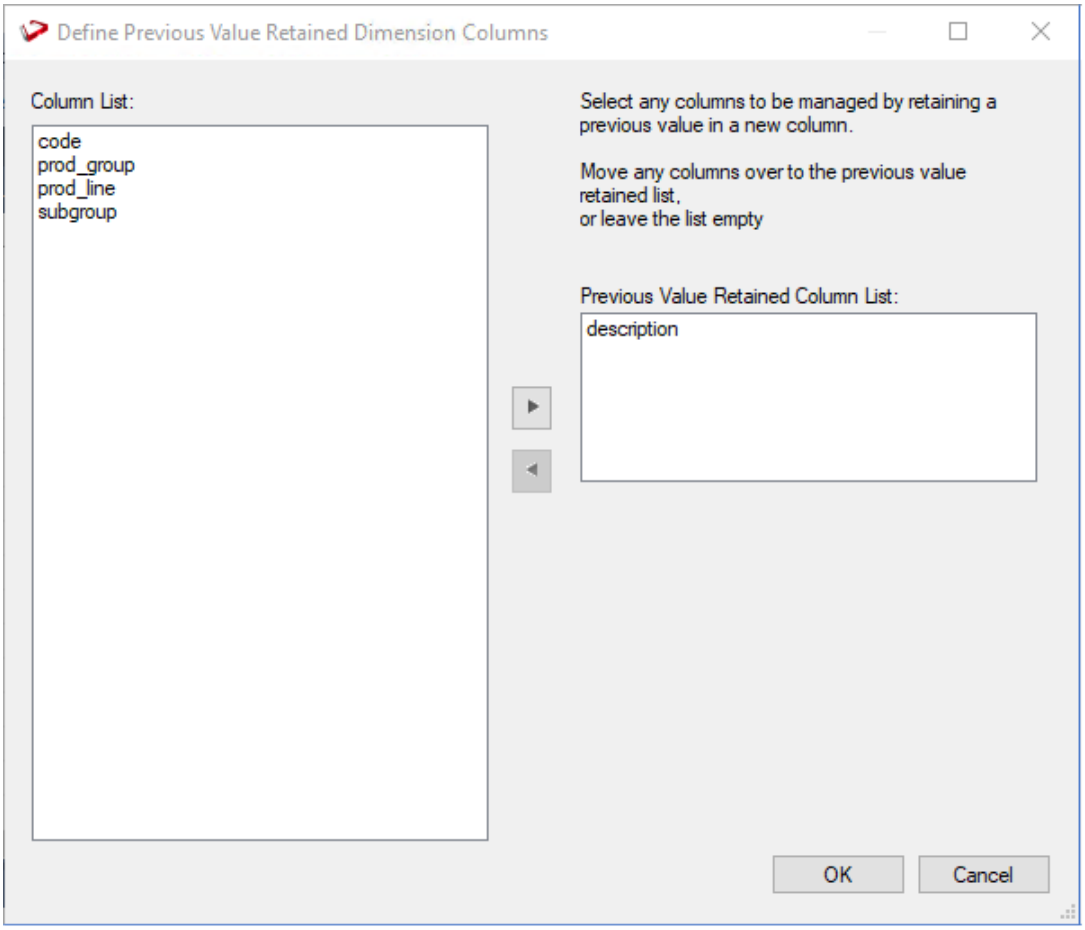
Tip

Do not over complicate the design of an analysis area. Keep it simple and avoid the unnecessary use of slowly changing dimensions.

Previous Values Dimension Type

If you selected a 'Previous values' dimension type then the following questions will be asked. If one of the other two types where chosen then proceed directly to **Dimension Properties**.

The following dialog appears:

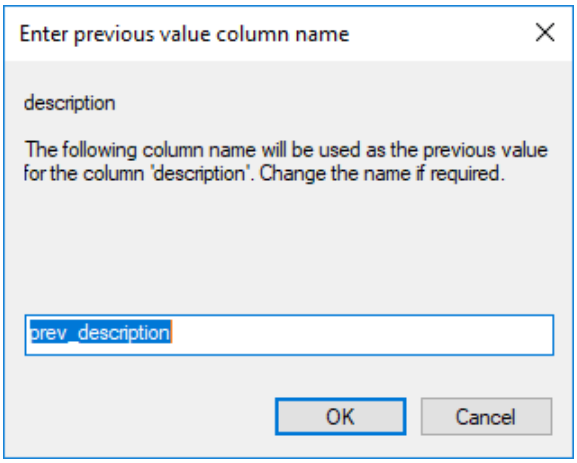


It requests the definition of each column that will be managed by storing the previous value in a separate additional column. Note that the business key cannot be used in this manner, as a change to the business key will result in a new record.

Select the columns to be managed and click **OK**.

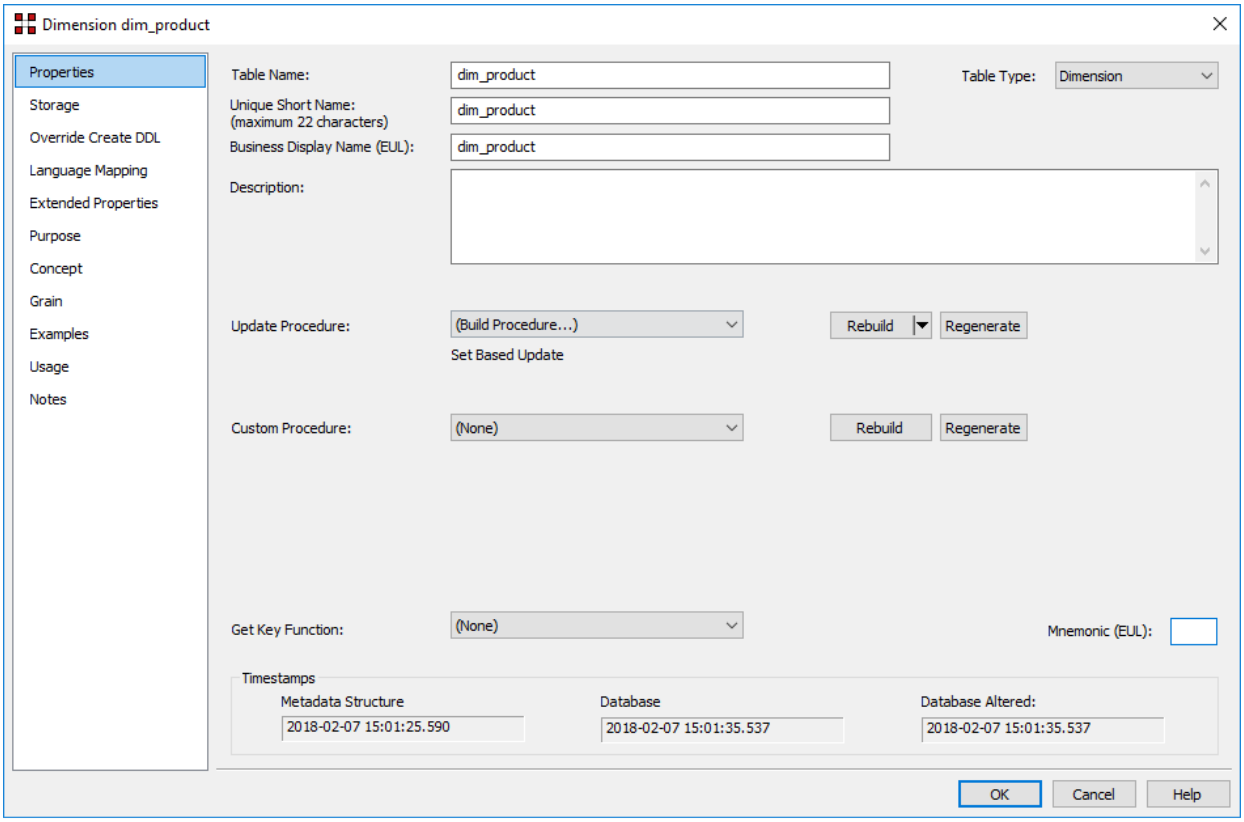
© Copyright, WhereScape Software Ltd All rights reserved.

The following dialog appears for each column that has been selected.
Either accept the name of the additional column or enter a new name.



Dimension Properties

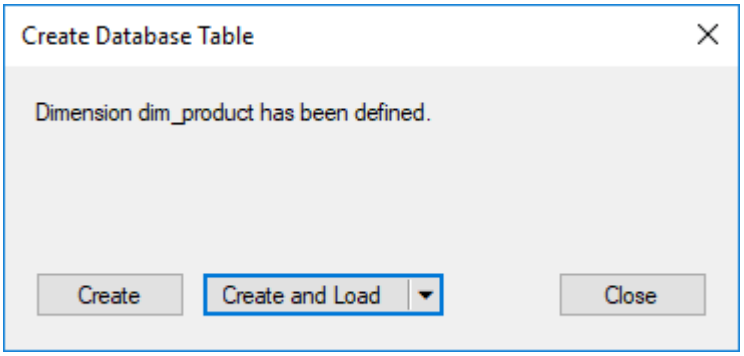
Once the dimension type is chosen, the **Properties** screen appears.
Change the storage options if required.
If prototyping, and the dimension is simple (i.e. one source table) then it is possible to create, load and update the dimension in a couple of steps.
If you wish to do this select the **(Build Procedure...)** option from the **Update Procedure** drop-down, and answer **Create and Load** to the next question.



© Copyright, WhereScape Software Ltd All rights reserved.

Create and Load

The following window appears after the **Properties** screen and asks if you want to create the dimension table in the database and execute the update procedure.



If you are satisfied with the columns that will be used and do not wish to add additional columns you can click the **Create and Load** button.

In this case it would be normal practice to select the **(Build Procedure...)** option from the 'Update Procedure' drop-down box in the previous **Properties** window.

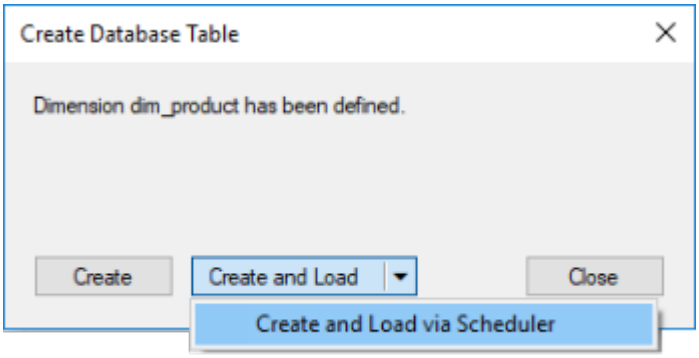
If no procedure has been selected then select the **Create** button as no update will happen in any event.

If **Create** or **Create and Load** is selected and a new procedure creation was chosen, proceed directly to the **Generating the Dimension Update Procedure** section.

If you have additional columns to add or columns to delete, click **Close** and proceed as follows below.

Note

It is possible to create and load the table via the Scheduler by selecting this option from the drop-down list on the **Create and Load** button:



Deleting and Changing columns

The columns defined for the dimension is displayed in the middle pane.

It is possible to delete any unwanted columns by highlighting a column name or a group of names and choosing the **Delete** key.

The name of a column can also be changed by selecting the column and using the right-click menu to edit its properties. Any new name must conform to the database naming standards. Good practice is to use alphanumerics and the underscore character.

Refer to the **Column Properties** for a more detailed description of the various fields.

Tip

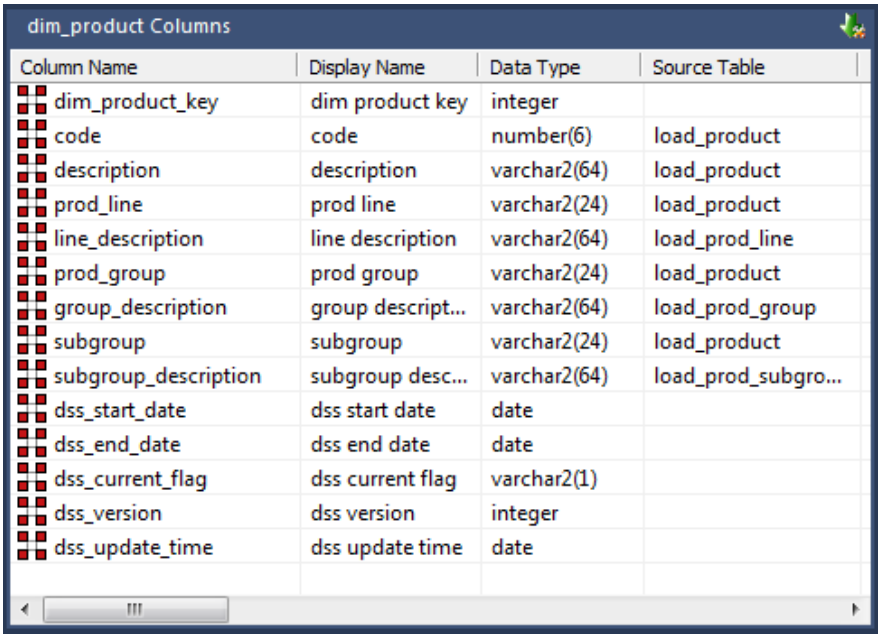
When prototyping, and in the initial stages of an analysis area build it is best not to remove columns, nor to change their names to any great extent. This type of activity is best left until after end users have used the data and provided feedback.

© Copyright, WhereScape Software Ltd All rights reserved.

Adding additional columns

With the columns of the dimension table displayed in the middle pane, this pane is considered a drop target for additional columns.

It is a simple matter therefore to select columns from other tables and to drag these columns into the middle pane.



The source table shows where each column was dragged from.

In the example above, the description column is acquired from the load_product, load_prod_group and load_prod_subgroup tables. To create the dimension table, these columns must be assigned unique names. For this example the last two columns have been renamed to group_description and subgroup_description.

There are a number of columns that do not have a source table. These columns have been added by WhereScape RED and are added depending on earlier choices.

A description of these columns follows:

Column Name	Description
dim_product_key	The unique identifier (artificial key) for the dimension. This key is used in the joins to the fact table. It is generated via a sequence (Oracle) or identity (SQL Server) associated with the table, except for the date dimension where it has the form YYYYMMDD
dss_start_date	Used for slowly changing dimensions. This column provides a date time stamp when the dimension record came into existence. It is used to ascertain which dimension record should be used when multiple are available.
dss_end_date	Used for slowly changing dimensions. This column provides a date time stamp when the dimension record ceased to be the current record. It is used to ascertain which dimension record should be used when multiple are available.
dss_current_flag	Used for slowly changing dimensions. This flag identifies the current record where multiple versions exist.
dss_source_system_key	Added to support dimensions that cannot be fully conformed, and the inclusion of subsequent source systems. See the ancillary settings section for more details.
dss_version	Used for slowly changing dimensions. This column contains the version number of a dimension record. Numbered from 1 upwards with the highest number being the latest or current version. It forms part of the unique constraint for the business key of a slowly changing dimension.
dss_update_time	Indicates when the record was last updated in the data warehouse.
dss_create_time	Indicates when the record was first created in the data warehouse

© Copyright, WhereScape Software Ltd All rights reserved.

Manually adding previous value columns

If a **Previous Value** type of dimension is chosen, or in fact if the dimension is converted to this type, it is possible to manually add any required columns that were not defined as part of the create. The steps are:

1. Add a new column by dragging in the column that is to have a previous value stored.
2. Change the name to a unique name. Typically by adding the prefix 'prev_' to the column name**.**
3. Change the source table, to be that of the dimension we are building.
4. Set the Key Type to 4.
5. Having performed these actions WhereScape RED will detect the column and build the appropriate code during the procedure generation phase.

Creating the table

Once the dimension has been defined in the metadata we need to physically create the table in the database.

1. To do this, right-click on the dimension name and choose **Create (ReCreate)** from the pop-up menu.
2. The **Results** pane show the results of the creation. It displays a message to the effect that the dimension table was created. A copy of the actual database create statement, the sequence create statement (Oracle only), and if defined the results of any index create statements will be listed. For the initial create no indexes will be defined.
3. If the table was not created then ascertain and fix the problem. A common problem is a 'Duplicate column' where a column has the same name in two of the source tables.
4. The best way to find such a column is to double-click on the list heading **Col name**; which will sort the column names into alphabetical order.
   Another double-click on the heading will sort the columns back into their create order.

For **Oracle** data warehouses, the sequence created as part of the table create is normally called **table_name_seq** unless the default has been overridden. If this sequence already exists, a warning is issued. This sequence is used in the assigning of a unique artificial key to the dimension. To change a sequence name while creating a dimension table, go to the **Storage** tab and change the **Artificial Key Sequence Name** field to the required sequence name. Refer to **Table Storage Screen - Oracle** for details.

SQL Server makes use of an identity attribute on the artificial key column.

The next section covers the **Generating the Dimension Update Procedure**.

© Copyright, WhereScape Software Ltd All rights reserved.

Generating the Dimension Update Procedure

Once a dimension has been defined in the meta data and created in the database, an update procedure can be generated to handle the joining of any tables and the update of the dimension records.

Notes

- You can also generate an update procedure via a template, refer to **Rebuilding Update Procedures** for details.
- If the dimension is created in a custom database target, then an additional processing option is available—PowerShell script-based processing. Refer to **Script Templates for Custom Database Table Objects** for details.

The zero key row

WhereScape RED by default, inserts a record into the dimension with an artificial key value of zero. This record is used to link any fact records that do not have valid dimension joins.

The values of the various columns in this record are acquired from the contents of the field **Zero Key Value** which is set in the Properties screen of each dimension column.

Generating a Procedure

1. To generate a procedure, right-click on the dimension name to edit the properties for the dimension.
2. From the **Update Procedure** drop-down list select **(Build Procedure...)**.
3. Click **OK** to update the properties and start the process of generating the new procedure.
4. A **Procedure Build Type** dialog appears, allowing you to select between a **Cursor** and **Set** procedure build types from the drop-down list.

A **Set** based procedure performs one SQL statement to join all the source tables together and then insert this data into the Dimension. This is normally the fastest method of populating a table.

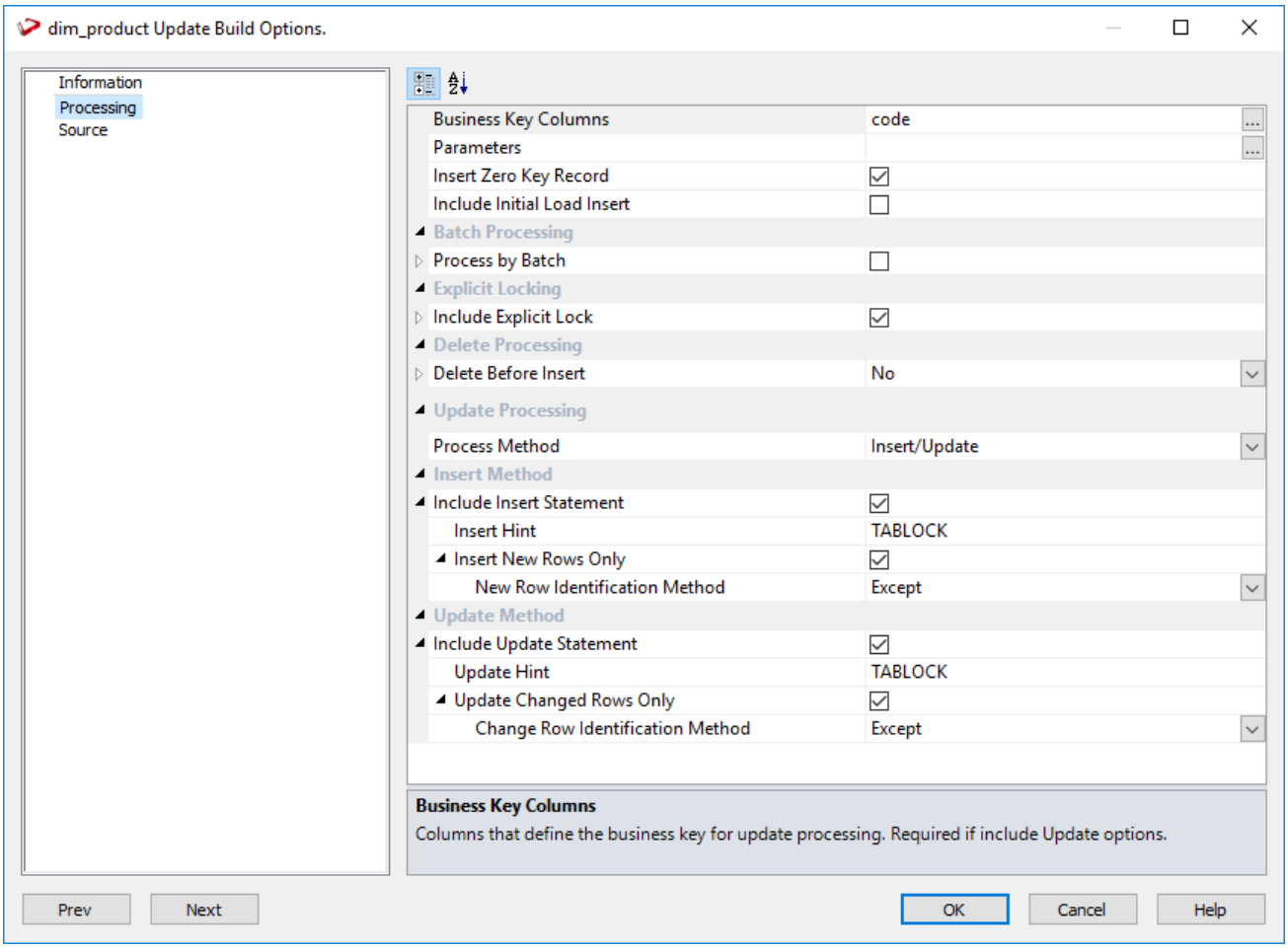
See below in this section for different options using the **Cursor** procedure building type.

Set Based Procedure Building types

1. Select **Set** from the drop-down menu.
2. The following window appears to enter the Update Procedure Build Options.
3. Click on the **ellipsis** button on the top rightmost corner to select the **Business Key Columns** for the dimension.

© Copyright, WhereScape Software Ltd All rights reserved.

Processing tab

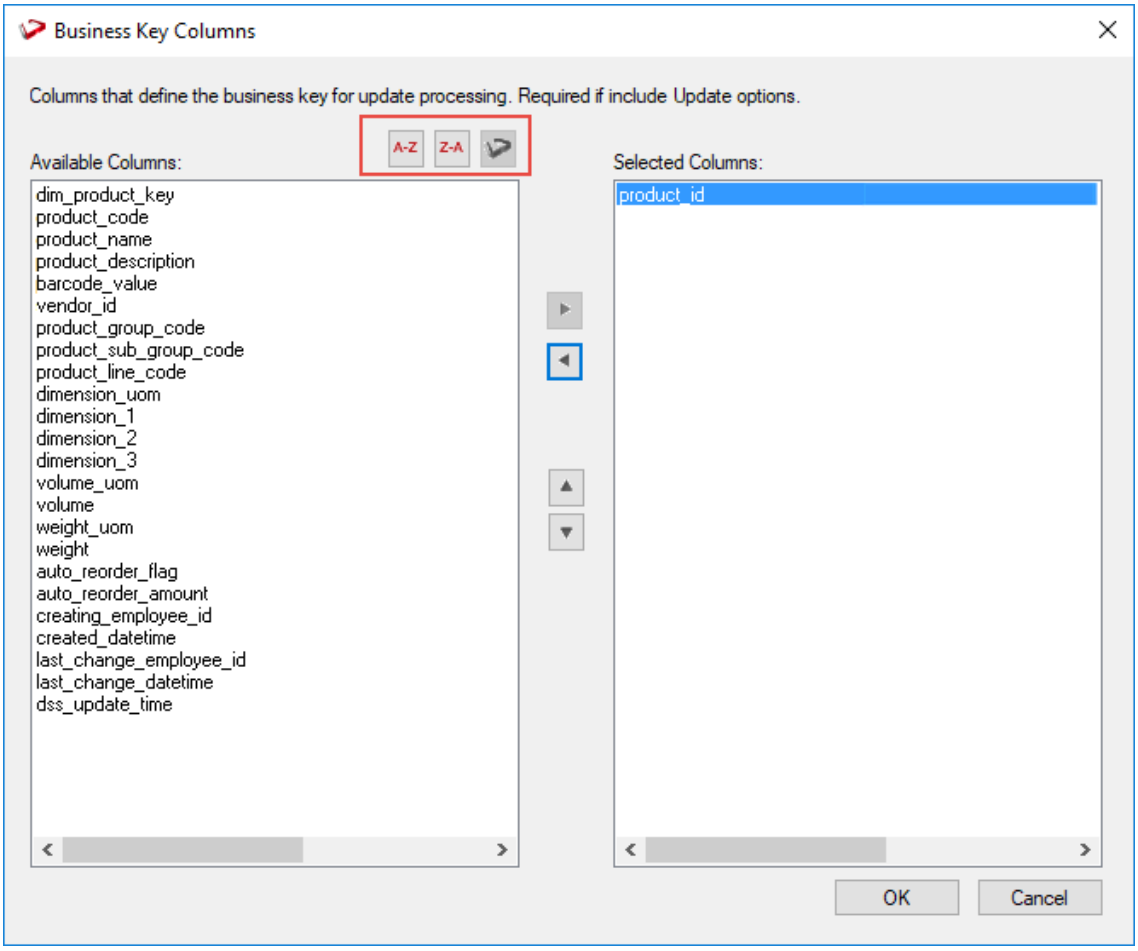


© Copyright, WhereScape Software Ltd All rights reserved.

Business key

A business key can be made up of multiple columns, but it must provide a unique identifier. Where multiple columns uniquely and separately identify the dimension, choose one to act as the primary business key.

For example, a source table may have a unique constraint on both a product code and a product description. Therefore, the description as well as the code must be unique. It is of course possible to combine the two columns, but the normal practice would be to choose the code as the business key.



Tips:

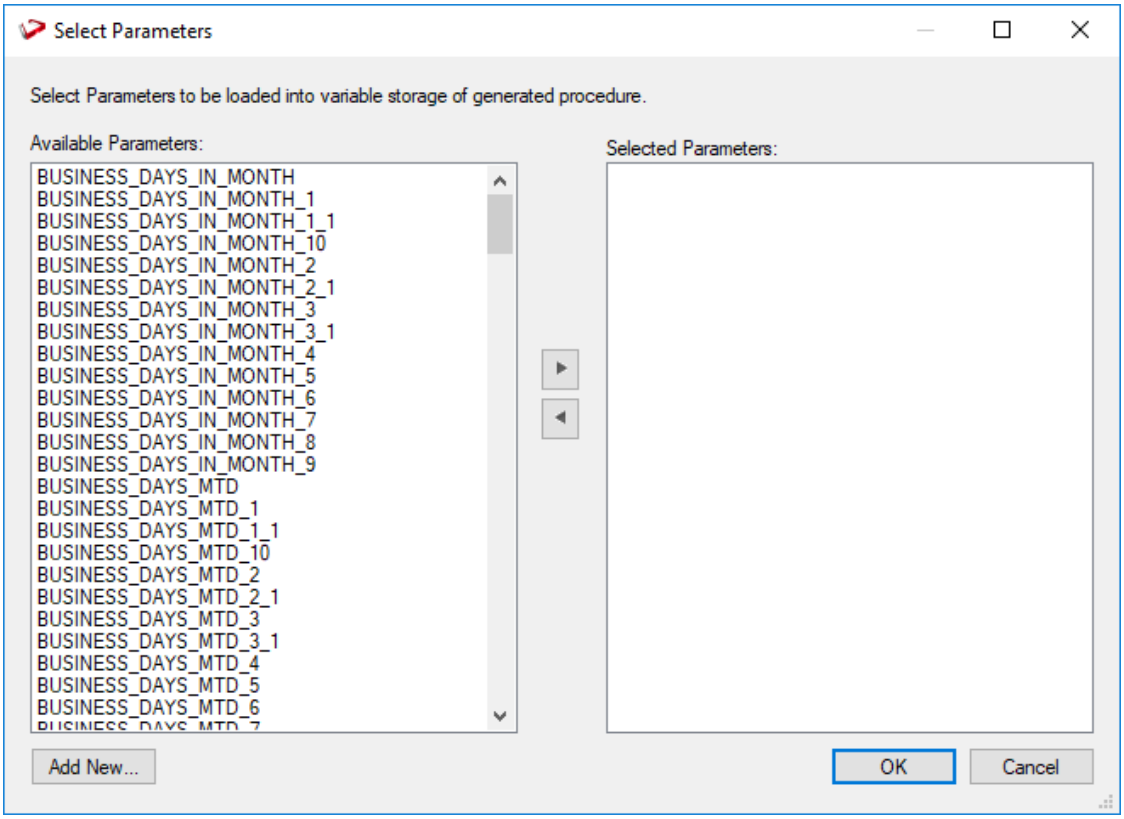
- Use the column name ascending/descending buttons to sort column names. To revert to the meta column order, click on the meta column order button.
- **NULL Values** - none of the columns chosen as the business key should ever contain a NULL value. See the note at the start of the Dimensions chapter.

© Copyright, WhereScape Software Ltd All rights reserved.

Parameters

If WhereScape RED parameters exist in the metadata, the following window is displayed. Any parameters selected in this window (by moving them to the right pane), are included in the generated update procedure as variables.

The procedure includes code to retrieve the value of the parameter at run time and store it in the declared variable.



The variables can also be used in column transformations and in the from/where clause for the update procedure. Some databases have a 30 character limit for variable names. WhereScape RED ensures the variables added for any parameters are less than 30 characters long by creating variable names in the form v_ followed by the first 28 characters of the parameter name.

For example, a parameter called MINIMUM_ORDER_NUMBER_SINCE_LAST_SOURCE_LOAD will be available as the variable v_MINIMUM_ORDER_NUMBER_SINCE_L.

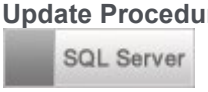
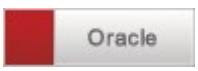
Tips

- WhereScape RED parameters should be unique within the first 28 characters to avoid conflicting variables names.
- If the desired parameter doesn't exist in the metadata yet, a new parameter can be added by clicking on the **Add New** button on the bottom leftmost corner of the Select Parameters dialog.

Refer to **Parameters** for more information about WhereScape RED Parameters.

Fields	Description
Enable Parallel DML	**Oracle Only** - Adds all the code required to the update procedure to enable Oracle parallel inserts. The default for this option is not set.
Insert Zero Key Record	This option enables you to include the zero key (unknown record) for tables with an artificial key. The default for this field is set.
Include Initial Load insert	This option adds an additional insert statement to the update procedure that runs if the target Dimension Object is empty. The benefit of this is improved performance inserting into an empty table without performing any checks to see if rows already exist. The default for this field is FALSE (i.e. an initial insert statement is not added

© Copyright, WhereScape Software Ltd All rights reserved.

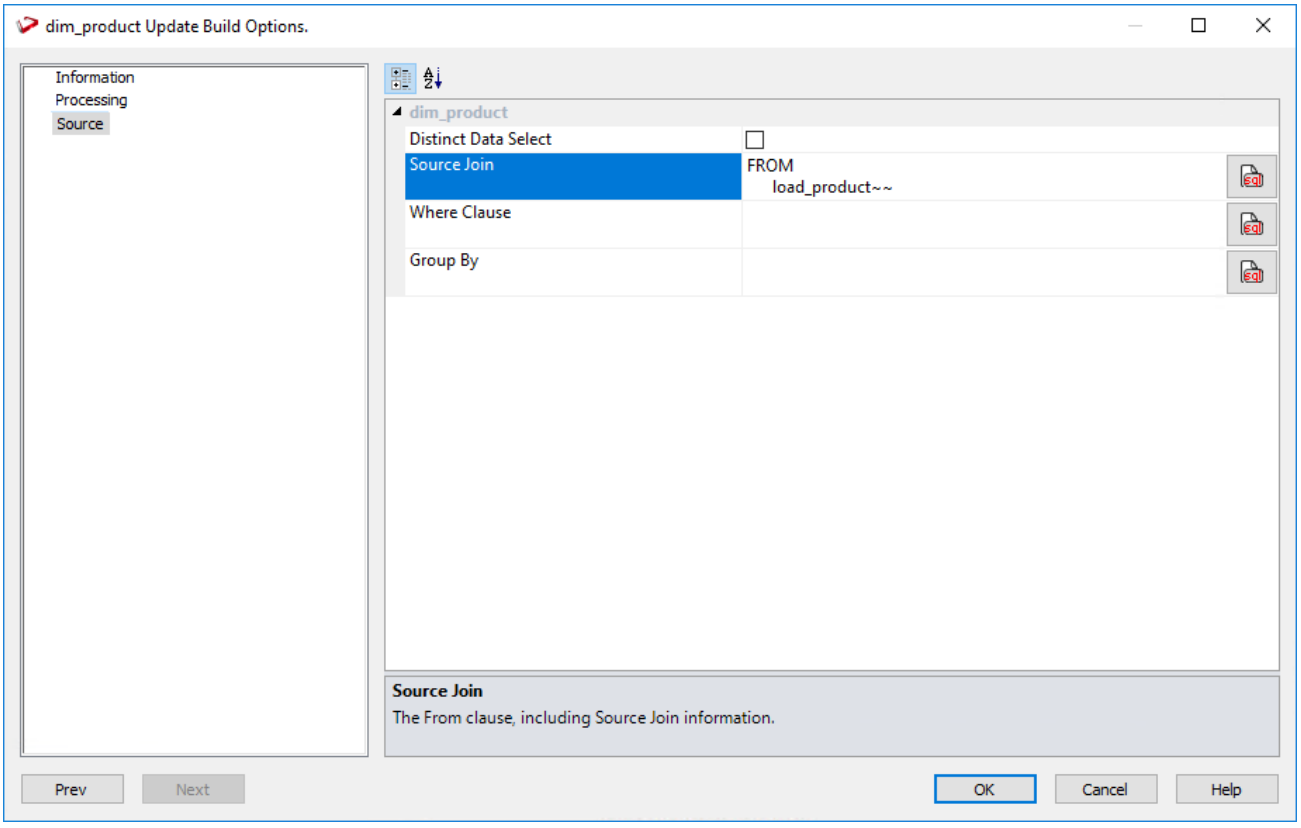
Fields	Description
	to the procedure).
Process by Batch	This field enables the user to select a column to break up the data being processed in a loop based on the distinct values in the field. The update procedure loops on this field and performs the delete, update and/or insert for each value. If the field chosen is a date datatype (date, datetime or timestamp), then the user is able to specify yearly, monthly, daily or column level looping. The default for this field is False (do not do batch processing).
Batch Processing Field	Enables selecting a field to batch process on. If you select a date field you will have the ability to process by date part. If you select a join field to process by you can choose and attribute of that related table to group by.
Include Explicit Lock	Enables locking of the table to avoid concurrent transactions. Double click the **Lock Clause** field to open a window that enables you to create the lock clause to insert.
Delete before insert	This option enables a delete statement to be added to the update procedure before any update or insert statement. This is a particularly useful option for purging old data and for updates based on a source system batch number. The default for this field is No which automatically grays out the **Issue Warning if a Delete Occurs** and the **Delete Where Clause fields.** • **Issue Warning if a Delete occurs** - sets the procedure to a warning state if deletes occur. • **Delete Where Clause** - The delete where clause is appended to the generated delete statement to constrain the rows deleted.
Process Method	Enables selecting whether the table should be updated using an Insert/Update statement or a Merge statement.
Source Table Locking	Enables a locking request modifier to be specified for each source table. The specified locking request modifier is applied to each source table during generated update procedures. By default this is set to 'ACCESS' which locks each row being accessed, a blank entry will result in no locking clause in the generated procedure. This option may also be presented in a separate dialog
Include Insert Statement	The Include Insert Statement option includes an insert statement in the procedure to insert new rows in the Dimension. If this option is set, the **Insert New Rows Only** option is available. If this option is turned off, the update procedure will not contain an insert statement. The default for this field is set (i.e. an insert statement is included).
Insert New Rows only	The insert new rows only option uses change detection to work out what rows require inserting. • **New Row Identification Method** - Method used to identify that the records in the source are not currently recorded in the target table.
Insert Hint	Enter a database hint to be used in the INSERT statement. This is an Oracle and SQL Server only option. Defaults can be configured in **Tools > Options > Default Update Procedure Options**.  Default is TABLOCK.  Default is APPEND.
Insert New Rows Only	Uses change detection to work out what rows require inserting.
New Row Identification Method	Method used to identify that records in source are not currently recorded in the target table.
Existing Data Selection Hint	Database-compliant hint to be used for the existing data select statement.

© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
Include Update Statement	Includes an update statement in the procedure to update changing rows in the Data Store. If this option is chosen, then the Update Changed rows only option is available.
Include Hint	Enter a database hint to be used in the INCLUDE statement. This is an Oracle and SQL Server only option. Defaults can be configured in **Tools > Options > Default Update Procedure Options**. SQL Server Default is TABLOCK. Oracle Default is APPEND.
Update Changed Rows Only	Uses change detection to work out what rows require updating. Choosing this option, enables the **Change Row identification Method**.
Change Row Identification Method	Method used to identify that records in source have changed from what is currently recorded in the target table.
Existing Data Selection Hint	Database-compliant hint to be used for the existing data select statement.
Merge Hint	Enter a database hint to be used in the MERGE statement. This is an Oracle and SQL Server only option. Defaults can be configured in **Tools > Options > Default Update Procedure Options**. SQL Server Default is TABLOCK. Oracle Default is APPEND.

Source tab

If only one source table was used, adding the **Business Key** and checking the above fields is enough to proceed, otherwise use the **Source** tab to **Join** the relevant tables.

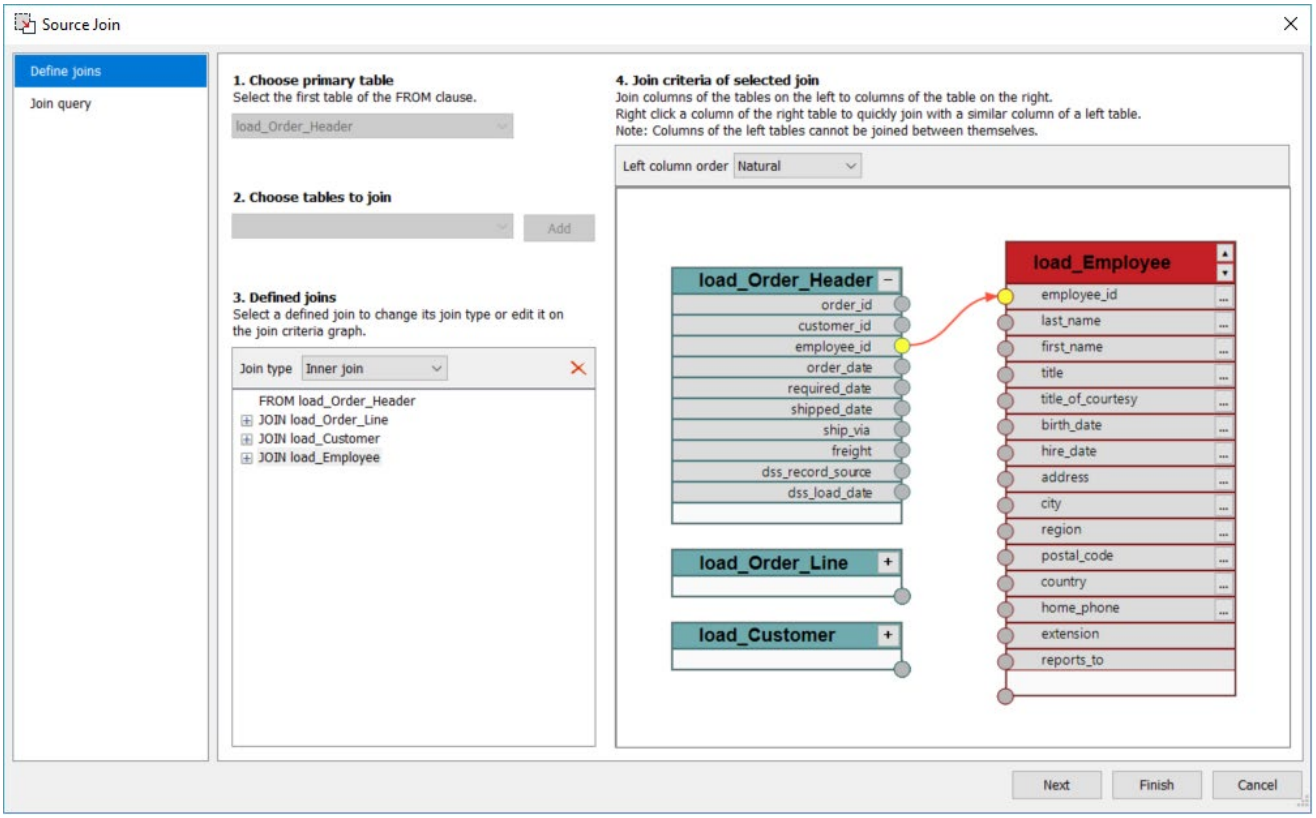


© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
Distinct Data Select	Ensures duplicate rows are not added to the Dimension. This is achieved by the word DISTINCT being added to the source select in the update procedure. The default for this field is not set.
Source Join	The From clause, including Source Join information. See example below for Joining multiple source tables.
Where Clause	The Where Clause. Use as a filter to extract only the necessary records that fulfill a specified criteria.
Group By	The Group by clause. Use in collaboration with the SELECT statement to arrange identical data into groups.

Joining multiple source tables

WhereScape RED provides a wizard that enables you to graphically generate the join clause for an object with multiple source tables. The **Source Join** wizard enables you to specify the primary (driving) table and the source tables to join. A graphical representation of the tables and the join criteria is displayed by the wizard which you can manipulate to change the defined joins.



The **Source Join** wizard guides you in generating the Join Clause through the following steps:

1. Choose Primary Table

   The first source table encountered is the default primary (driving) table which can be changed before any joins are defined.
2. Choose Tables to Join

   Once the primary table is set, the remaining source tables are listed in this drop down. Select a table to join and then click **Add**. The two tables are added to pane 4 and the basis for the join is displayed in pane 3.
3. Defined Joins

   The join type defaults to **Inner** join but this can be changed from the drop down.

© Copyright, WhereScape Software Ltd All rights reserved.

4. Join criteria of selected join

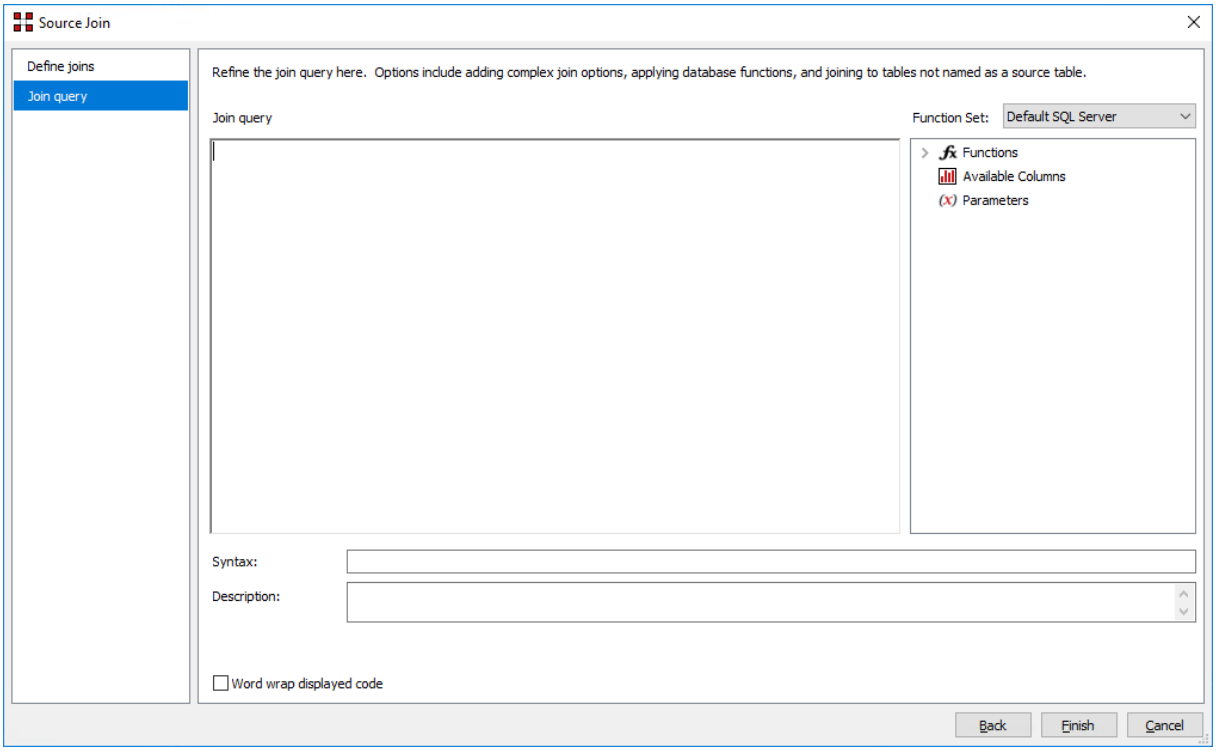
   The column(s) used to join the two tables can be joined manually by clicking the corresponding join column connection point from the left-hand table and then dragging the line that appears to the connection point of the required column in the table on the right.

   Alternatively, if a column with a matching or similar name exists in the left hand table(s) then a context menu is provided on the columns on the right-hand table. You can right click on the column to display the context menu and then select a column from the list.

Tip

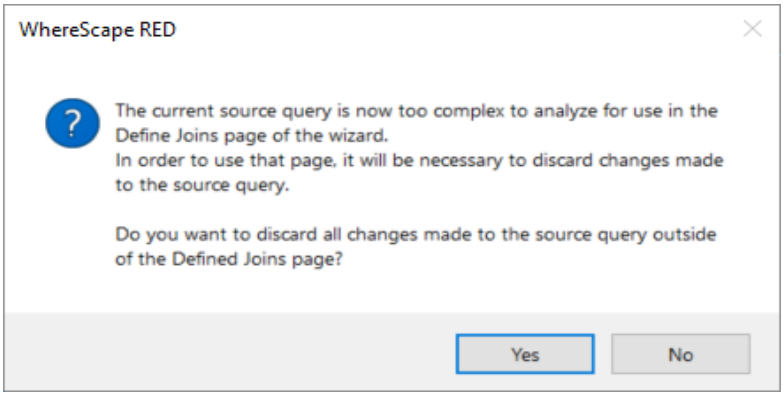
To aid manual column join, you can expand or collapse columns in the left-hand tables and toggle between natural and alphabetic column ordering.

5. Once the join column(s) for the table pair are defined, then any additional source tables can be added from the drop-down list on step 2 and the above column mapping process repeat to join this table. This process is repeated until all source tables are joined.
6. Clicking the **Next** button or the **Join query** tab progresses to the join editor window, where custom joins, database functions, etc. can be added.



Note

If RED is unable to parse a join statement, the user cannot navigate back to the **Define joins** tab and the following warning is displayed:



© Copyright, WhereScape Software Ltd All rights reserved.

Change Detection tab

For Slowly Changing Dimension columns. If the dimension was defined as a Slowly Changing Dimension, the **Change Detection** tab is displayed in the table **Update Build Options** screen.

Note:

For custom database targets, you can set the initial default values for the DSS columns that are used in the procedure generation for slowly changing dimensions.
Please refer to the section **DSS Columns for Custom Targets** for details. The default values set are populated in the corresponding fields of the **Change Detection** tab.

A change detection field(s) must be selected for this Dimension type.

Fields	Description
Change Detection Fields	Click the ellipsis button to select the change detection fields that are required for the dimension
Null Support	If the Add Null support is selected, the change detect column management caters for null values in any change detect columns. If this option is not selected and null values are present, errors may occur running the update procedure. The default for this value is ON (i.e. null values are catered for).
Update Current Records Only	The update current record only option only applies changes to non-change detect columns on the current record. If this option is not selected, all past (non-current) rows will also be updated to reflect changes to non-change detect columns. The default for this value is ON (i.e. only the current record is updated).
Reset Dates to Initial Values	Resets dss_start date and dss_end_date date values to original values.
Start Date for Initial Member	The start date for initial member field contains the start date for the first version of a particular business key. The value should be specified in an appropriate form, taking into account the default date format in the databases. The date may need to be explicitly cast to the current data type. The default value provided is usually cast to the correct database and can be treated as a template. The default for this field is 1 January 1900.
End Date for Current Member	The end date for current member field contains the end date for the current version (the row with a current flag of Y and the maximum version number) of a particular business key. The value must be specified in an appropriate form, taking into account the default date format in the databases. The date may need to be explicitly cast to the current data type. The default value provided is usually cast to the correct database and can be treated as a template. The default for this field is 31 December 2999.
Start Date for New Member Entry	The start date for new member entry field contains the start date for any subsequent rows added to the history table (not the first row for a particular business key i.e. not version 1). The value must be specified in an appropriate form, taking into account the default date format in the databases. The date may need to be explicitly cast to the current data type. The default value provided is usually cast to the correct database and can be treated as a template. The default for this field is the current date and time.
End Date for Expiring Member Entry	The end date for the expiring member entry field contains the end date for any rows updated no longer to no longer be the current row in the history table (i.e. rows that are replaced by a new current row). The value must be specified in an appropriate form, taking into account the default date format in the databases. The date may need to be explicitly cast to the current data type. The default value provided is usually cast to the correct database and can be treated as a template. The default for this field is the current date and time less an arbitrary small amount (for SQL Server this is 0.00000005 of a day, or about 4 thousandth of a second).

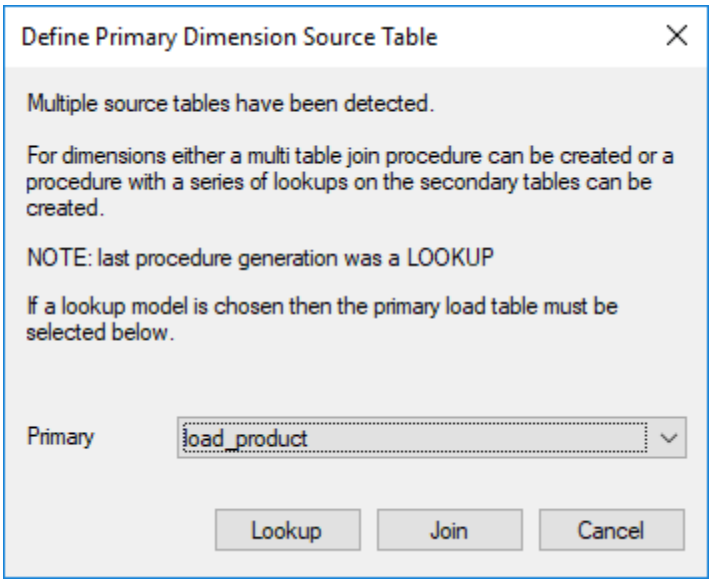
© Copyright, WhereScape Software Ltd All rights reserved.

Cursor procedure building types

Source Join / Lookup

If columns from multiple source tables were used to create the Dimension with a **Cursor** Procedure building type, the window below prompt users to Define the Primary Dimension source table.

The fields need to be joined in the Source Join field.



The choice provided is either to create a join that combines all the information from the various tables into one large select statement, or to do a series of lookups.

The normal process would be to join the tables to create one large select statement. There are however situations where this is not desirable or possible. When a large number of tables are involved then a join is slower than a series of lookups. In some situations, we may not have all the information to do an initial join. A series of lookups may be required to build up the information.

For complex situations, it is necessary to edit and enhance the generated procedure code.

If Lookup is chosen, then the primary table for the Dimension must be chosen. The philosophy here is that a series of lookups is conducted against the secondary tables using information from this primary table.

© Copyright, WhereScape Software Ltd All rights reserved.

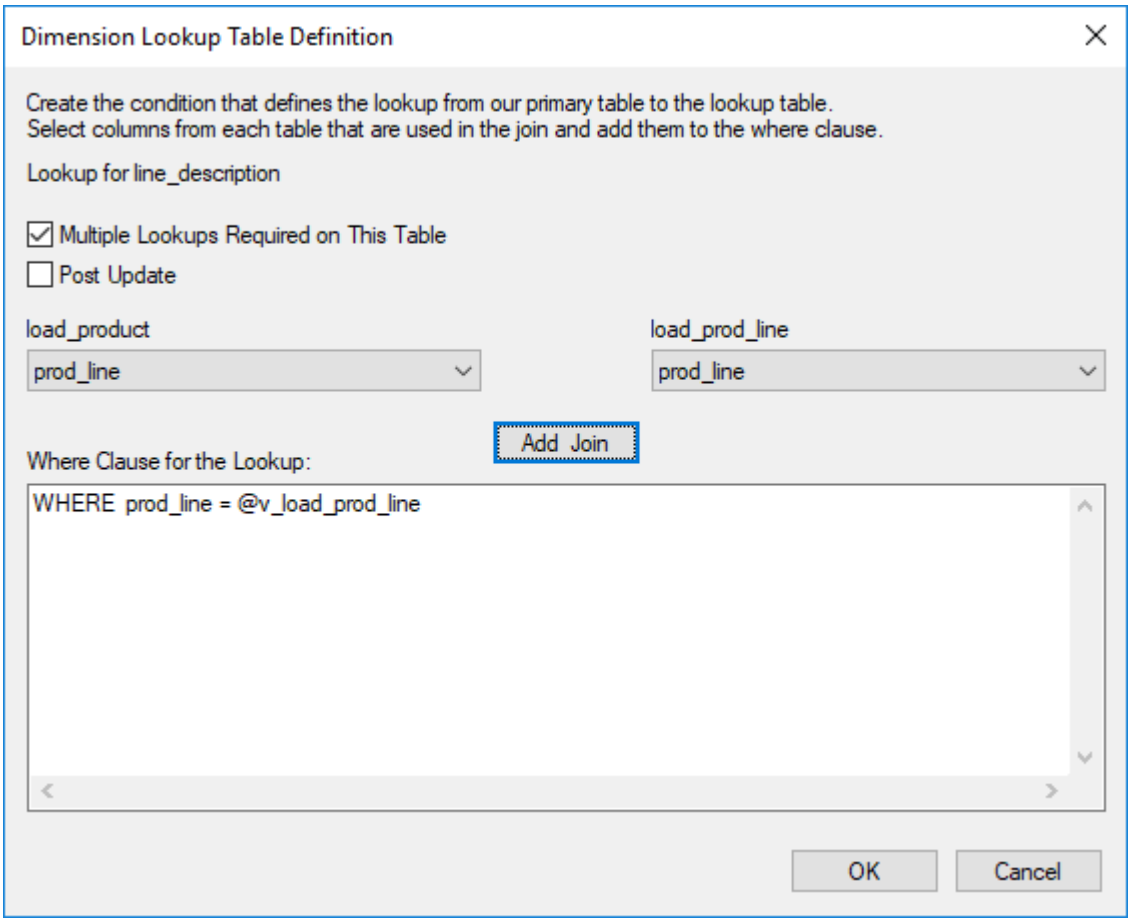
Lookup

If multiple source tables were used to build the Dimension, then an option would have appeared in an earlier window to choose between a join or a lookup.

If **Lookup** was chosen, then a window appears prompting for the joins between the primary table and each secondary (lookup) table.

The load_product table is the main table and a lookup is being performed on the load_prod_line table. Columns are selected from the drop-down column list for each table and then the **Add Join** button clicked to add these two columns to the 'Where' clause as shown in the bottom box.

This 'Where' clause may be edited and goes to form a **lookup** select statement within the main cursor loop of the update procedure.
In this example the 'Where' clause has had the second line manually added to identify the type of lookup in the code table.

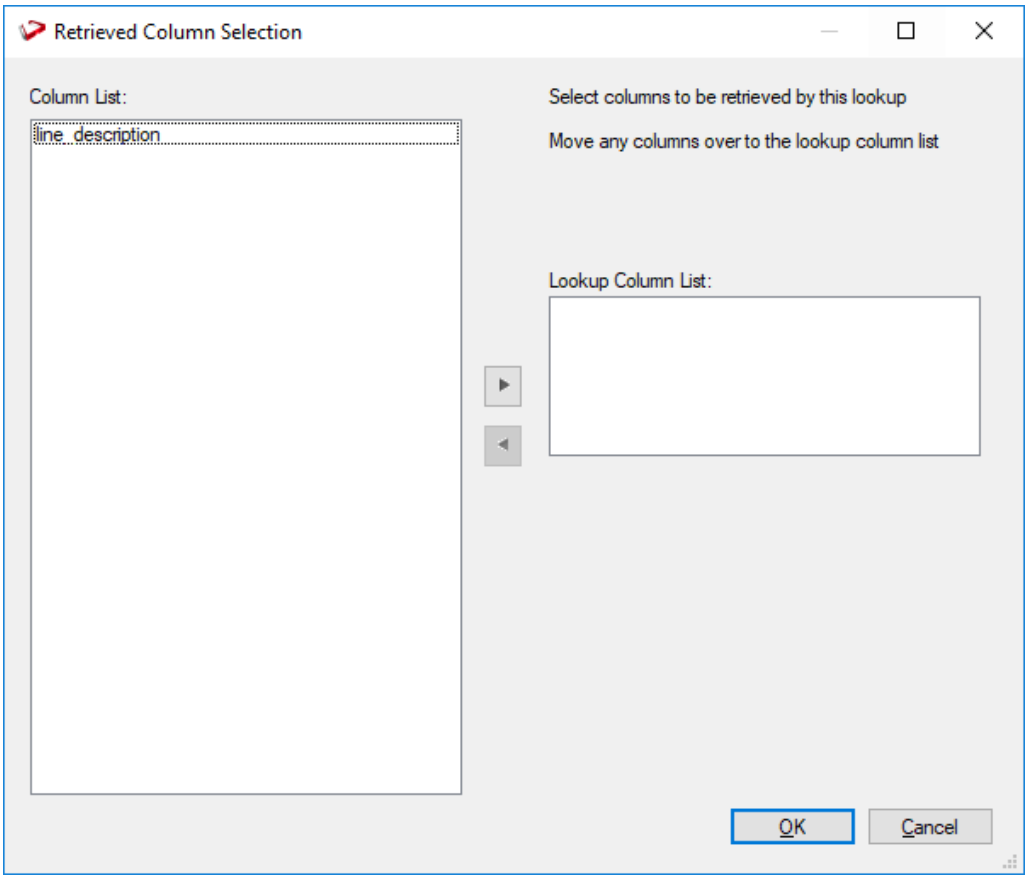


Multiple lookups can occur on the same table. Where we have a generic code table, for example, we may do multiple lookups to get descriptions.

The check-box **Multiple Lookups Required on This Table** results in repeated lookups against the second table until all columns sourced from this table have been utilized in the lookup statements.

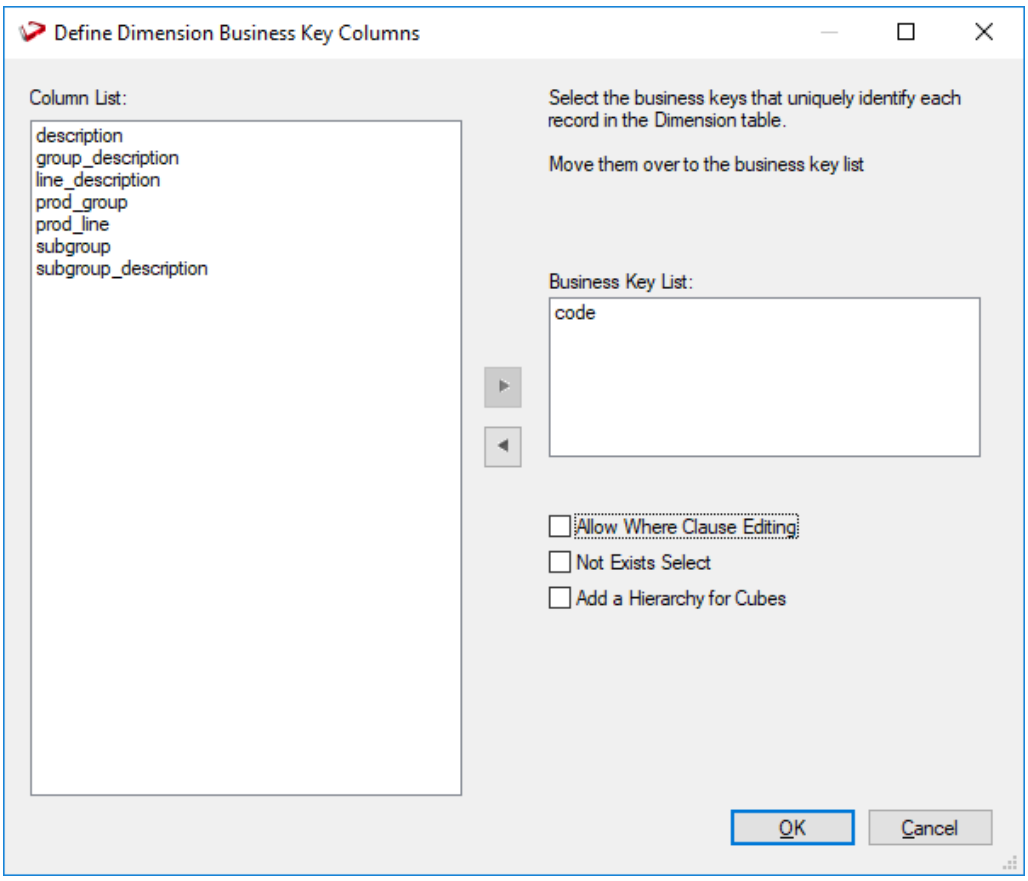
If the **Multiple Lookup** check-box is selected, then the following window appears to enable selection of the columns that are to be retrieved by the lookup.

© Copyright, WhereScape Software Ltd All rights reserved.



Business Key

Define the **Business Key** Columns by adding one or more business key columns in the right pane of the **Define Dimension Business Key Columns** window.



© Copyright, WhereScape Software Ltd All rights reserved.

Allow Where Clause Editing

- **Not Exists Select** - if this check-box is selected, it adds additional code to the main select statement. This code excludes every row in the source table that has not changed in the Dimension table and can result in faster execution of the update procedure for large Dimensions.
- **Add a hierarchy for Cubes** - this check-box opens another window that enables the definition of a simple hierarchy structure. This is useful if the hierarchy of the Dimension is known and the Dimension is used in the creation of Analysis services cubes. Hierarchies may be defined now or later when required. They may also be defined or modified after the dimension has been created.

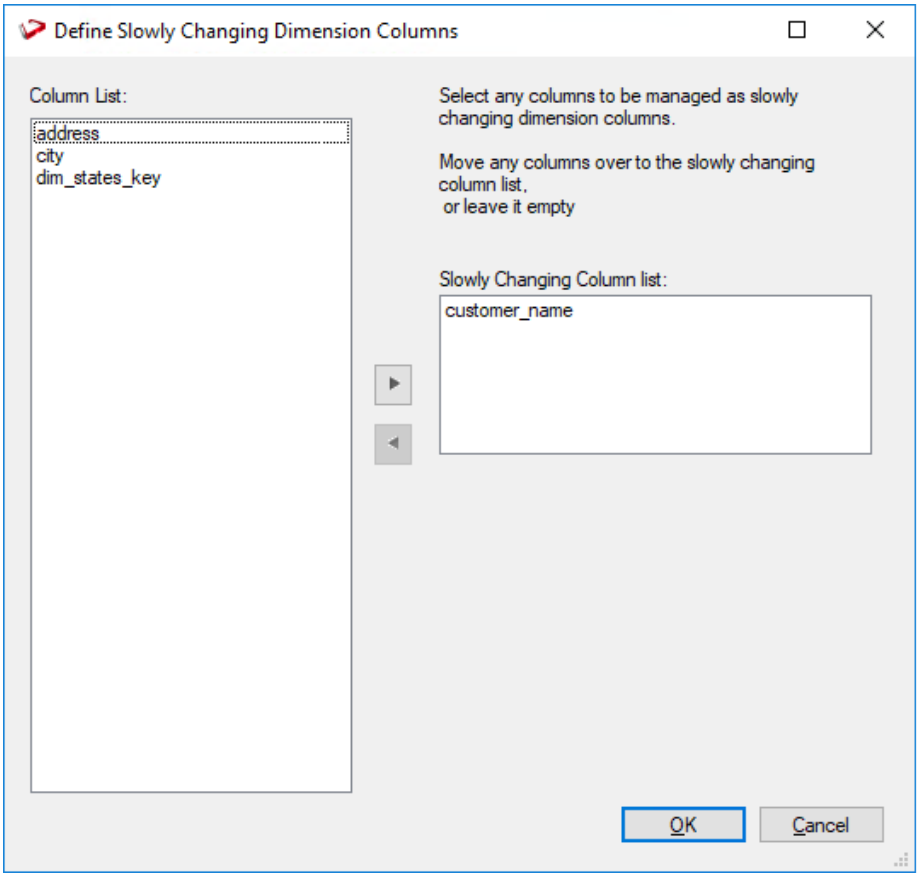
Slowly Changing columns

If the Dimension was defined as a slowly changing dimension, then the following additional window appears.

The first requests the selection of the columns to be managed as Slowly Changing Dimension columns.

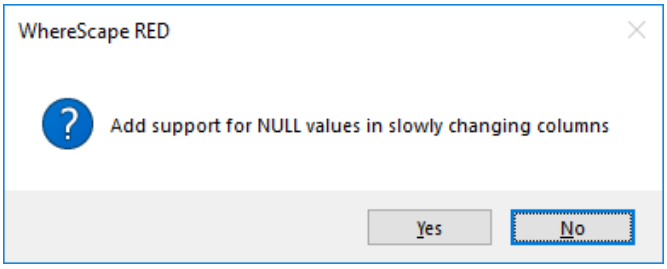
Select the required columns and then click **OK** to proceed.

In the example below, the customer name is to be managed as a Slowly Changing column in a customer Dimension.



The advantages and disadvantages of Slowly Changing Dimensions are discussed earlier in this chapter, but as a general rule try to minimize their usage, as they invariably complicate the processing and end user queries.

The following window appears if one or more columns are chosen for management.



© Copyright, WhereScape Software Ltd All rights reserved.

NULL values are the enemy of a successful data warehouse. They result in unreliable query results and can often lead to a lack of confidence in the data.

If a column is considered important enough to be managed as a Slowly Changing column, then it should not normally contain null values. It is often best to ensure that a Null cannot occur by using a IsNull() (SQL Server), Nvl() (Oracle) or Coalesce() (DB2) transformation when loading the column.

If however, this situation is unavoidable then answering **Yes** to this question results in additional code to test for Nulls, during the update of the Dimension records.

If **No** is answered and Nulls are encountered then a unique constraint violation occurs during the Dimension update.

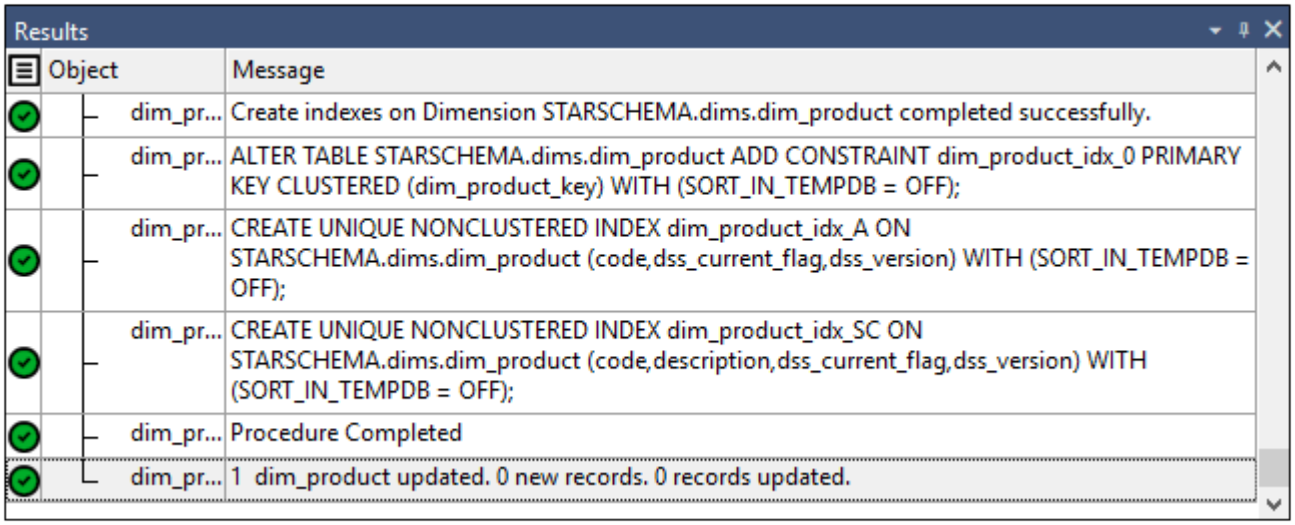
Building and Compiling the Procedure

- Once the above questions are completed the procedure is built and compiled automatically.
- If the compile fails an error is displayed along with the first few lines of error messages. Compile fails typically occur when the physical creation of the table was not done.
- If the compile fails for some other reason, the best approach is to use the Procedure editor to edit and compile the procedure. The procedure editor highlights all the errors within the context of the procedure.
- Once the procedure has been successfully compiled, it can either be executed interactively or passed to the RED Scheduler.

Indexes

By default, a number of indexes is created to support the Dimension. These indexes is added once the procedure has been built.

An example of the type of indexes created is as follows:



This example shows three indexes created:

1. A primary key constraint placed on the artificial key for the Dimension.
2. A unique index placed on the business key for the Dimension.
3. A unique index placed on the business key and a Slowly Changing column from the Dimension.

This third index is only created when a **Slowly Changing** Dimension is chosen.

Additional indexes can be added, or these indexes changed. Refer to **Indexes** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

Dimension Artificial Keys

The artificial (surrogate) key for a dimension is set via a sequence in Oracle and an identity column in SQL Server and DB2.

This artificial key normally, and by default, starts at one and progresses as far as is required.

A WhereScape standard for the creation of special rows in the dimension is as follows:

Key value	Usage
1 upwards	The normal dimension artificial keys are numbered from 1 upwards, with a new number assigned for each distinct dimension record.
0	Used as a join to the dimension when no valid join existed. It is the normal convention in the WhereScape generated code that any dimension business key that either does not exist or does not match is assigned to key 0.
-1 through -9	Used for special cases. The most common being where a dimension is not appropriate for the record. For example, we may have a sales system that has a promotion dimension. Not all sales have promotions. In this situation, it is best to create a specific record in the dimension that indicates that a fact table record does not have a promotion. The stage table procedure would be modified to assign such records to this specific key. A new key is used rather than 0 as we want to distinguish between records that are invalid and not appropriate.
-10 backward	Pseudo records. In many cases we have to deal with different granularities in our fact data. For example, we may have a fact table that contains actual sales at a product SKU level and budget information at a product group level. The product dimension only contains SKU based information. To be able to map the budget records to the dimension, we need to create these pseudo keys that relate to product groups. The values -10 and backwards are normally used for such keys. A template called 'Pseudo' is shipped with WhereScape RED to illustrate the generation of these pseudo records in the dimension table.

Dimension Get Key Function

When WhereScape RED generated a new procedure to support the update of a Dimension, it also generates a **Get Key** function (procedure in SQL Server and DB2). This function is also generated when Dimension Views are created and can be generated for retro-fitted Dimensions.

The **Get Key** function is used to return an artificial key when supplied a business key. The normal syntax is to call the function passing the business key and be returned a status indicating the result of the lookup.

In a successful lookup the artificial key for the dimension record is also returned. If the lookup fails because the business key is not found in the Dimension, then an appropriate status is returned, and the artificial key is set to 0.

Notes:

- Two parameters exist on the generated **Get Key** function that allow manipulation of the process. It is possible to automatically add a new Dimension record when a match is not made. It is also possible to record the lookup failure in the detail/error log. By default, both these options are disabled.
- The **Get Key** function is not available for Dimension objects that are stored in custom database targets.

Modifying the 'Get Key' function

There may be a situation where a Dimension record can have multiple business keys.

For example:
You may be able to look up a 'factory' based on the factory code, the name of the factory or its address. The generated **Get Key** function only supports the primary business key which in this case may be the factory code. However, we may have a fact record that only contains the factory name. In such a case, we need to modify the **Get Key** function or add a second function. The advantages of adding a second function are that

© Copyright, WhereScape Software Ltd All rights reserved.

any code generated in the future that uses this Dimension works without modification. The disadvantages are the multiple points of maintenance required when changing the Dimension. Both options are discussed below:

Adding a second 'Get Key' function for a dimension

Choose a name similar to the existing function. If say we have a 'get_dim_factory_key' function, then we may choose a name like 'get_factory_name_key'. Proceed to create and populate the function as follows:

Right-click the **Procedure** group in the left pane and select **New Object** to create a new procedure. Enter the name and click **ADD**.

1. A Properties window is displayed. Change the type to **Function** for Oracle, leave as a **Procedure** for SQL Server. Enter a description under the **Purpose** column and click the **UPDATE** button.
2. Expand the **Procedure** group in the left pane and double-click the new object just created. This opens the procedure editor window.
3. Select the **Tools > View Procedure or Template** menu option.
4. Select the existing **Get Key** function from the procedure drop-down. A new window appears on the right, showing the existing **Get Key** function.
5. Click in the existing function window. Right-click and choose **Select All** and then right-click again to select **Copy**.
6. Click in the main editor window and paste the existing function into the window. The existing function window can now be closed if desired.
7. Using the menu option **Edit > Replace**, change all occurrences of the old function name to the new name.
8. Change the various occurrences of the business key name and variable to match the new one and test it all.

Once a new **Get Key** function has been produced, it is necessary to modify the Stage update procedure to utilize the new function.

Tip:

An alternative approach is to create a Dimension View, specifically to generate a different kind of get key function. Then use this view's surrogate key in the Stage table instead of the Dimension table's surrogate key. This approach avoids modifying the staging table procedure.

Extending the 'Get Key' function

If you want to keep all the information for the lookup of dimension keys in one location, then extend the functionality of the existing **Get Key** function. The normal practice in this case is to include all potential business keys as parameters. Multiple select statements can be employed to select the keys based on the appropriate business key. To choose the correct statement either use Nulls in the unused business keys or include an additional parameter to indicate which business key is being passed.

If this method is used, then any Stage update procedures must be modified to handle the new calling syntax.

Dimension Initial Build

The initial population of a dimension with data can be achieved by generating a custom procedure and then use the right-click context menu to select the **Execute Custom Procedure via Scheduler** option.

The dimension should be analyzed once the custom procedure is completed so that the database query optimizer can make use of the indexes.

For smaller dimensions (e.g. less than 500,000 rows) run the normal **Update** procedure against an empty dimension table. There is however a danger in this action in that the query optimizer will not have any information on the table, and hence will do a full table pass when adding each new row. For a very small dimension this will not be an issue, but it will rapidly become a major problem as the row count increases.

The problem with the initial update is that the database does not know to use the index that has been created for the business key, and hence does a full table pass when attempting to update/insert a record.

To prevent this problem, the generated code will issue an analyze statement after 1000 rows have been added to the table. The statement used is as follows:

SQL Server: update statistics *dim_table* **with sample 5 percent**
Oracle: analyze table *dim_table* **estimate statistics sample 5 percent for all indexed columns**

© Copyright, WhereScape Software Ltd All rights reserved.

DB2: runstats on table *dim_table*
where *dim_table* is the name of the dimension.

This command is issued whenever 1000 or more new rows are added to a dimension.

If this is not required then the code should be removed from the update procedure. It may not be required if a plan has been locked in for the table.

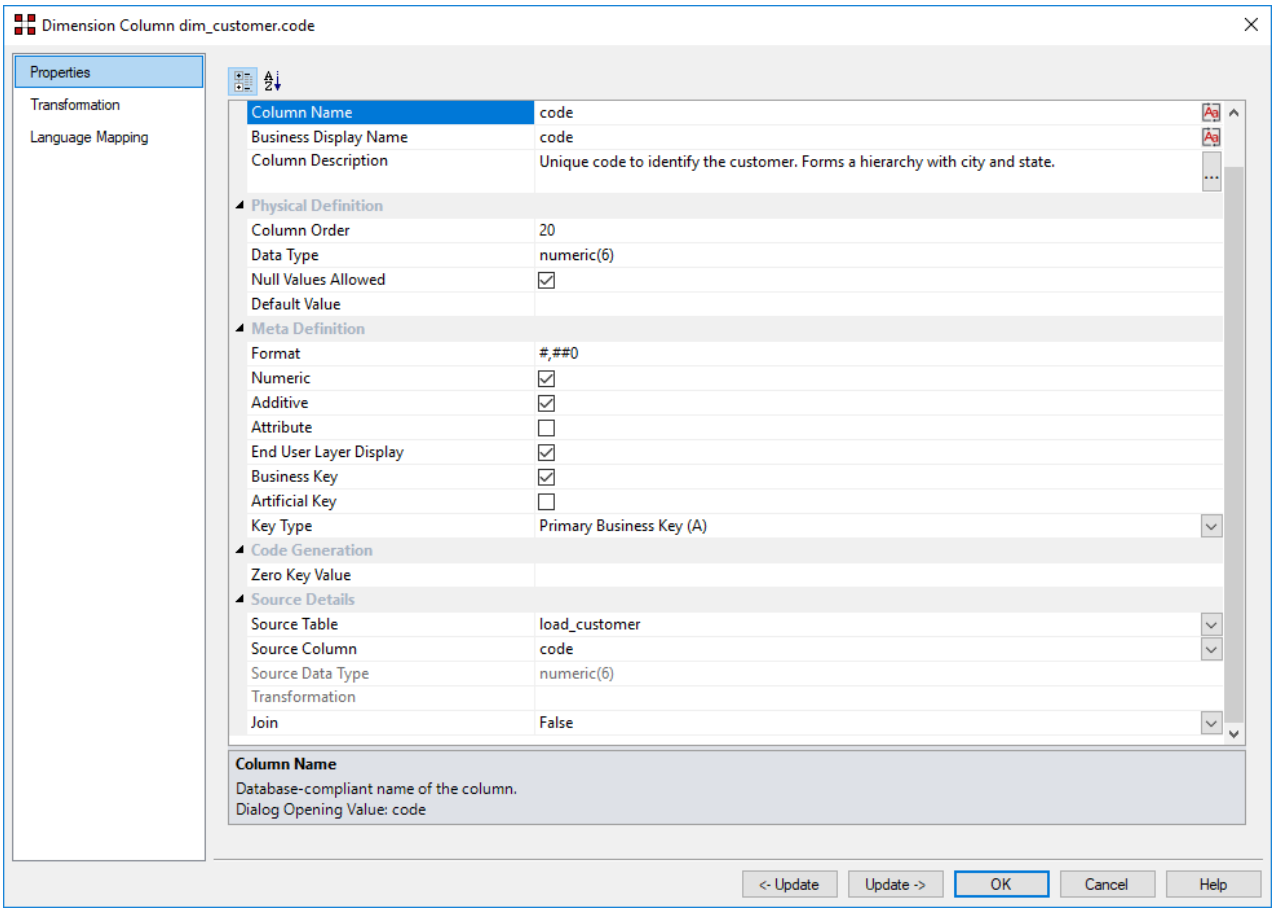
Dimension Column Properties

Each dimension column has a set of associated properties. The definition of each property is described below:

Tip:

If a database table's definition is changed in the metadata, then the table needs to be altered in the database. Use the **Validate > Validate Table Create Status** to compare metadata definitions to physical database tables.
The option also provides the ability to alter the database table, through a pop-up menu option from the validated table name. See the example below.

A sample **Properties** screen is as follows:



Tip:

The two special update keys enables you to update the column and step either forward or backward to the next column's properties.
ALT-Left Arrow and **ALT-Right Arrow** can also be used instead of the two special update keys.

General

Options	Description
Table Name	Database-compliant name of the table that contains the column. [Read-only].

© Copyright, WhereScape Software Ltd All rights reserved.

Options	Description
Column Name	Database-compliant name of the column. Typically column-naming standards exclude the use of spaces etc. A good practice is to only use alphanumerics, and the underscore character. Changing this field alters the table's definition. **Note** A case conversion button on the right converts the text between different cases: UPPER CASE, Capitalized Case and lower case. The mode cycles to the next case in the sequence each conversion.
Business Display Name / Column Title	Name that the business uses to refer to the column, which is included in the RED-generated documentation and can be used in the end user layer of other tools. [Does NOT affect the physical database table]. As such it is a free form entry and any characters are valid. **Note** A case conversion button on the right converts the text between different cases: UPPER CASE, Capitalized Case and lower case. The mode cycles to the next case in the sequence each conversion.
Column Description	This field contains the description for the column. It may be a description from a business user's point of view. This field might additionally contain information on where and how the column was acquired. For example, if the column is sourced from multiple tables or is a composite or derived column then this definition would normally describe the process used to populate the column. This field is used in the documentation and is available via the view **ws_admin_v_dim_col** . This field is also stored as a comment against the column in the database.

Physical Definition

Options	Description
Column Order	Numeric value that controls the relative order of columns in the database create statement. The lowest numbered column will appear first in the table. Although this affects the physical table definition no action will be taken unless the table is re-created in the database. The columns can be re-numbered based on the existing order by choosing the **Respace Order Number** pop-up menu option when positioned over any column in the table. This action will number the columns in increments of 10 starting at 10. In addition to a simple change of the order field, the column order can be changed by first displaying the columns in the middle pane and then using drag and drop to move the columns around. This drag and drop process will automatically renumber the columns as required.
Data Type	Database-compliant data type that must be valid for the target database. Typical data types for Oracle are integer, number, char(), varchar2() and date. For SQL Server, common types are integer, numeric, varchar() and datetime. See the database documentation for a description of the data types available. Changing this field alters the table's definition.
Null Values Allowed	Determines whether the table column can hold NULL values or whether a value is always mandatory.
Default Value	Initial value that is assigned to the column when a row is inserted into the table but no value is specified for the column.
Character Set	Teradata Only Database-compliant table column character-set used for storage.
Format	Teradata Only Database-compliant table column format. It does not affect the physical table definition, but rather provides input to the view ws_admin_v_dim_col which can be used to assist in the population of an end user tool's end user layer. As such it is a

© Copyright, WhereScape Software Ltd All rights reserved.

Options	Description
	free form entry and any characters are valid. Typically format masks are only used on numeric fields. Example: #,###0.00. It is not worth the effort of populating this field unless it can be utilized by the end user tools in use.
Character Comparison/Sorting	Teradata Only Determines how the column character values are treated for comparison and sorting operations. Choose from: case specific, not case specific, uppercase case specific or uppercase not case specific.
Compress/Compress Value	Teradata Only Optional list of values to be compressed. By default, only NULL is compressed if no list of values is specified.

Meta Definition

Options	Description
Format	Optional format mask that can be used in end user tools. [Does NOT affect the physical database table]. As such it is a free form entry and any characters are valid. Typically format masks are only used on numeric fields. Example: #,###0.00. It is not worth the effort of populating this field unless it can be utilized by the end user tools in use.
Numeric	Indicates whether the table column holds values that are numeric. This is normally only relevant for fact tables. It does not affect the physical table definition, but rather provides input to the view **ws_admin_v_dim_col** which can be used to assist in the population of an end user tool's end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.
Additive	Indicates whether the table column holds values that are additive. This implies that the column can be summed when performing data grouping in a query. This is normally only relevant for fact tables. It does not affect the physical table definition, but rather provides input to the view **ws_admin_v_dim_col** which can be used to assist in the population of an end user tool's end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.
Attribute	Indicates whether the table column holds values that are descriptive, and/or are used for grouping/summing. An attribute is defined as a column that is non factual. For example we may have an order number, or a invoice number stored in the fact table. Such columns are considered attributes, rather than facts. This checkbox is therefore normally only relevant for fact tables. This check box does not affect the physical table definition, but rather provides input to the view **ws_admin_v_dim_col** which can be used to assist in the population of an end user tools end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.
End User Layer Display	Indicates whether the table column is available/visible to end users. If set the documentation will include the column in the glossary and in the user documentation. It is also used to decide what columns appear in the view **ws_admin_v_dim_col**. Typically columns such as the artificial key would not be enabled for end user display.
Business Key	Indicates whether the column is part of the BUSINESS key, which is defined during the update procedure build. [Normally maintained automatically]. Multiple columns can form the primary business key.
Artificial Key	Indicates whether the column is the generated artificial/surrogate key (unique identifier) for the table. Only one artificial key per table is supported. [Normally maintained automatically].
Key Type	Key type that is assigned and used when generating the table's update procedure and indexes. [Normally maintained automatically]. It can be altered here, but this should only be done if the consequences are fully understood and tested.

© Copyright, WhereScape Software Ltd All rights reserved.

Options	Description
	The supported values are: **Key type** / **Meaning** **0** — The artificial key. Set when the key is added during drag and drop table generation. **1** — Component of all business keys. Indicates that this column is used as part of any business key. For example: By default the dss_source_system_key is added to every dimension table. It is considered part of any lookup on that table and has the key type set to 1. Set when the column is added during drag and drop table generation. **2** — Indicates that this column is a dimensional join. Used on fact tables to indicate the dimension keys. Results in an index being built for the column (Bitmap in Oracle). Set during the update procedure generation for a fact table, based on information from the staging table. **3** — Slowly changing column indicator. Used on dimension tables to indicate that the column is being managed as a slowly changing column within the context of a slowly changing dimension. Set when a column is identified during the dimension update procedure generation. **4** — Slowly changing column indicator. Used on dimension tables to indicate that the column is being managed as a slowly changing column within the context of a slowly changing dimension. Set when a column is identified during the dimension update procedure generation. **5** — Start date of a date ranged dimension. Used on dimension tables to indicate that the column is defined as the starting date for a source system date ranged dimension. Forms part of the business key. Set during the dimension creation. **6** — End date of a date ranged dimension. Used on dimension tables to indicate that the column is defined as the ending date for a source system date ranged dimension. Forms part of the business key. Set during the dimension creation. **A** — Indicates that the column is part of the primary business key. Set whenever a business key is defined as part of an update procedure generation. **B-Z** — Indicates that the column is part of a secondary business key. Only used during index generation and not normally set.

Code Generation

Options	Description
Zero Key Value	Determines the value populated for the column in the **Invalid Join** or **Unknown record**. By default, NULL is used when a value is not specified. All dimensions that use standard WhereScape RED generated procedures have a row with an artificial key of zero. This row is used to link to the fact records when no match on the dimension can be found. For example we may have some sales records that come through without a product defined. WhereScape RED will be default associate these fact records with the zero key product dimension entry. So, we might set the zero key value to 'Unknown product' for the name column in the product dimension.

© Copyright, WhereScape Software Ltd All rights reserved.

Source Details

Options	Description
Source Table	Identifies the source table where the column's data comes from. This source table is normally a load table, or a dimension table within the data warehouse. If the column was sourced from multiple tables, then the normal practice is to record one of the tables in this field and a comment listing all of the other tables in the **Source strategy** field. This field is used when generating a procedure to update the stage table. It is also used in the track back diagrams and in the documentation.
Source Column	Identifies the source column where the column's data comes from. Such a column is normally a load table column, which in turn may have been a transformation or the combination of multiple columns. This may also be a dimensional key where a dimension is being joined.
Source Data Type	Identifies the source column's data type. [Read-only].
Transformation	Refer to **Dimension Column Transformations** for details. [Read-only].
Join	Indicates whether the table column is used in a table join. [Normally maintained automatically but can be optionally changed to override the default join logic used in the generated update procedure]. The **Source table** and **Source column** fields will provide the dimension table's side of the join. The options for this field are: False, True, Manual and Pre Join. • Setting this field to Manual changes the way the Dimension table is looked up during the Stage table update procedure build. It enables you to join the dimension manually in the Cursor mapping dialog (where the 'Where' clause is built). The usual dialog for matching the dimension business key to a column or columns in the Stage table is not displayed, if this option is enabled. • Setting this field to Pre Join activates the **Pre Join Source Table** field and enables you to select a table from the drop-down list.

Changing a Column Name

If the **Column name** or **Data type** is changed for a column then the metadata will differ from the table as recorded in the database.

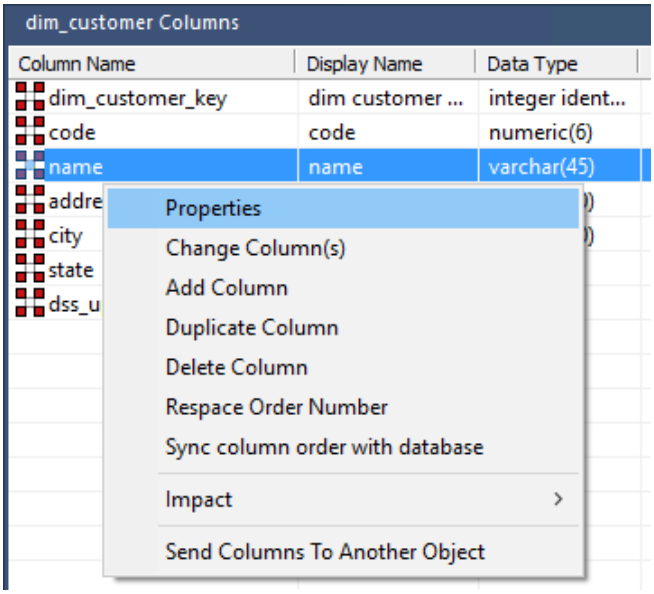
- Use the **Validate > Validate Table Create Status** menu option or the right-click menu to compare the metadata to the table in the database.
- A right-click menu option of **Alter Table** is available when positioned on the table name after the validate has completed. This option will alter the database table to match the metadata definition.

For example: Analysis Services does not like **name** as a column name.

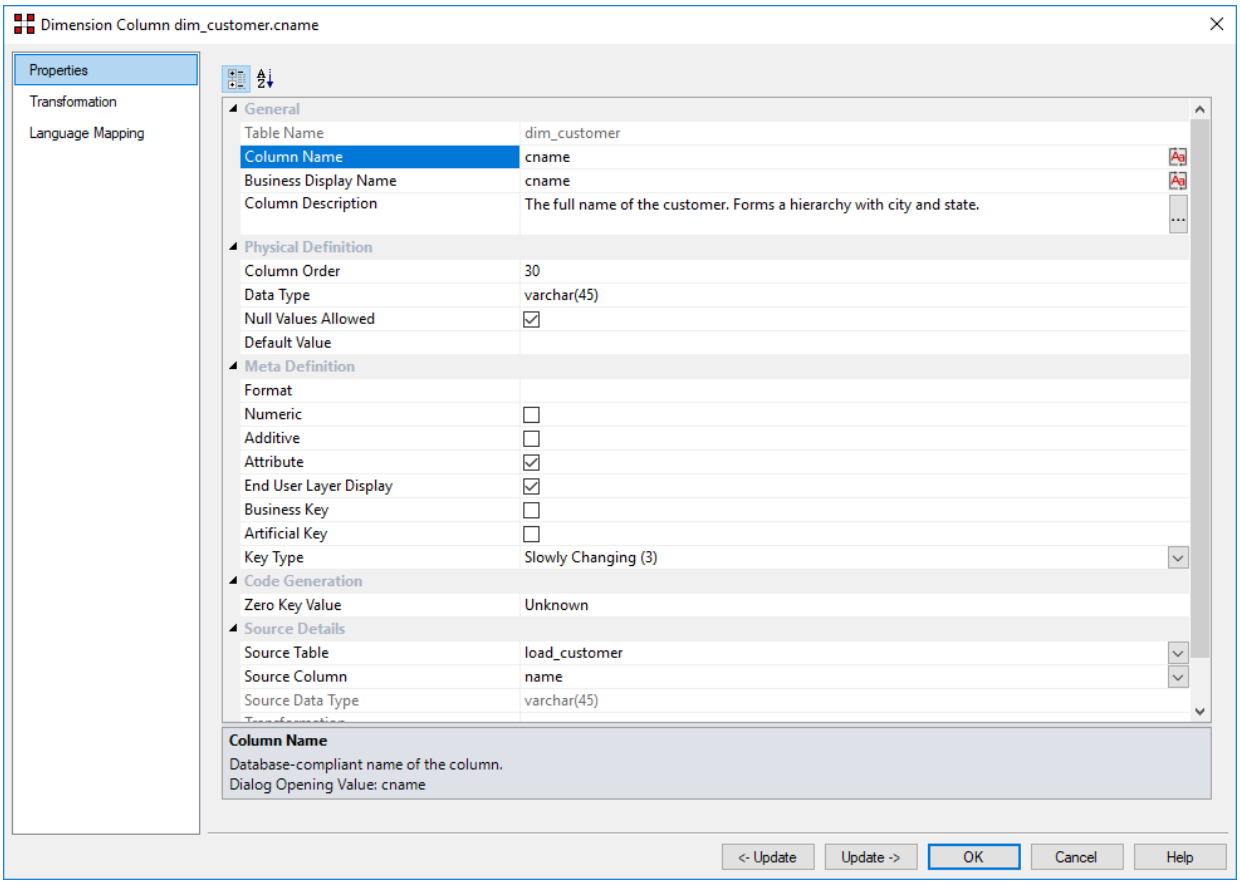
For dim_customer it will therefore be necessary to change the column name from **name** to **cname**.

1. Click on the **dim_customer** object in the left pane to display the dim_customer columns in the middle pane.
2. When positioned on the column **name** in the middle pane, right-click and select **Properties** from the drop-down menu.

© Copyright, WhereScape Software Ltd All rights reserved.

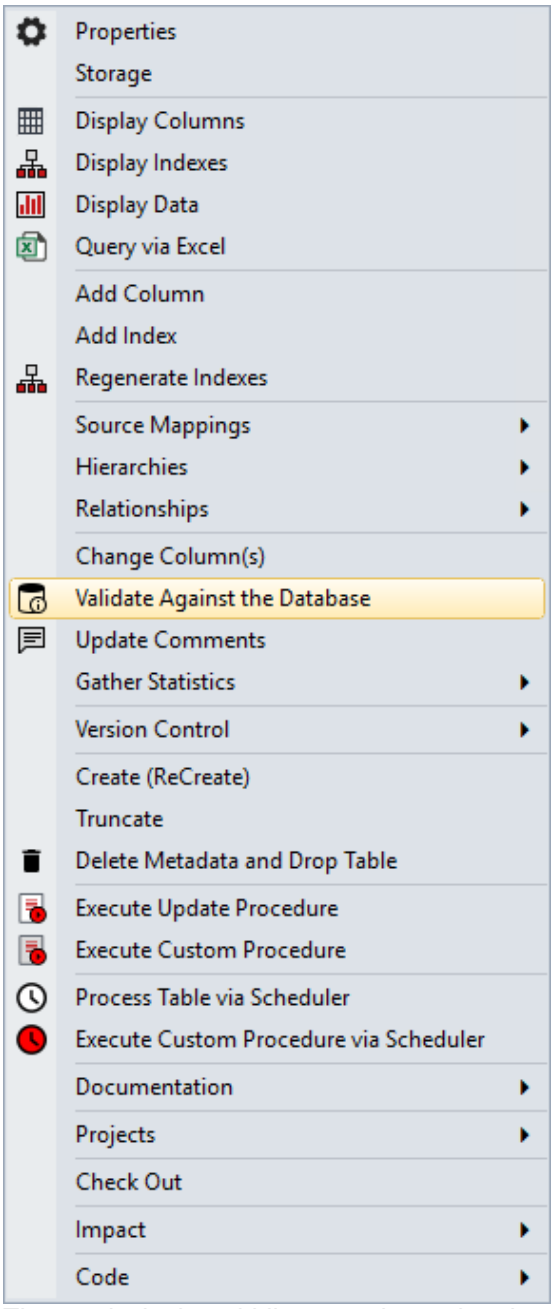


3. Change the column name from **name** to **cname** as shown below. Click **OK** to leave the **Properties** window.

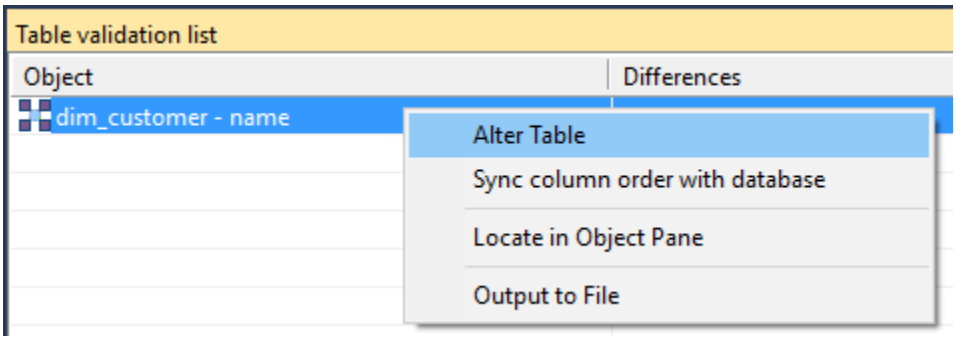


© Copyright, WhereScape Software Ltd All rights reserved.

4. Right-click the **dim_customer** object in the left pane and select **Validate against Database**.



5. The results in the middle pane shows that the metadata has been changed to **cname** while the column name in the database is still **name**.
6. Right-click **dim_customer** in the middle pane and select **Alter table** from the context menu.



7. A warning appears, displaying the table and column name to be altered. Select **Alter Table**.
8. A dialog appears confirming that dim_customer has been altered. Click **OK**.

© Copyright, WhereScape Software Ltd All rights reserved.

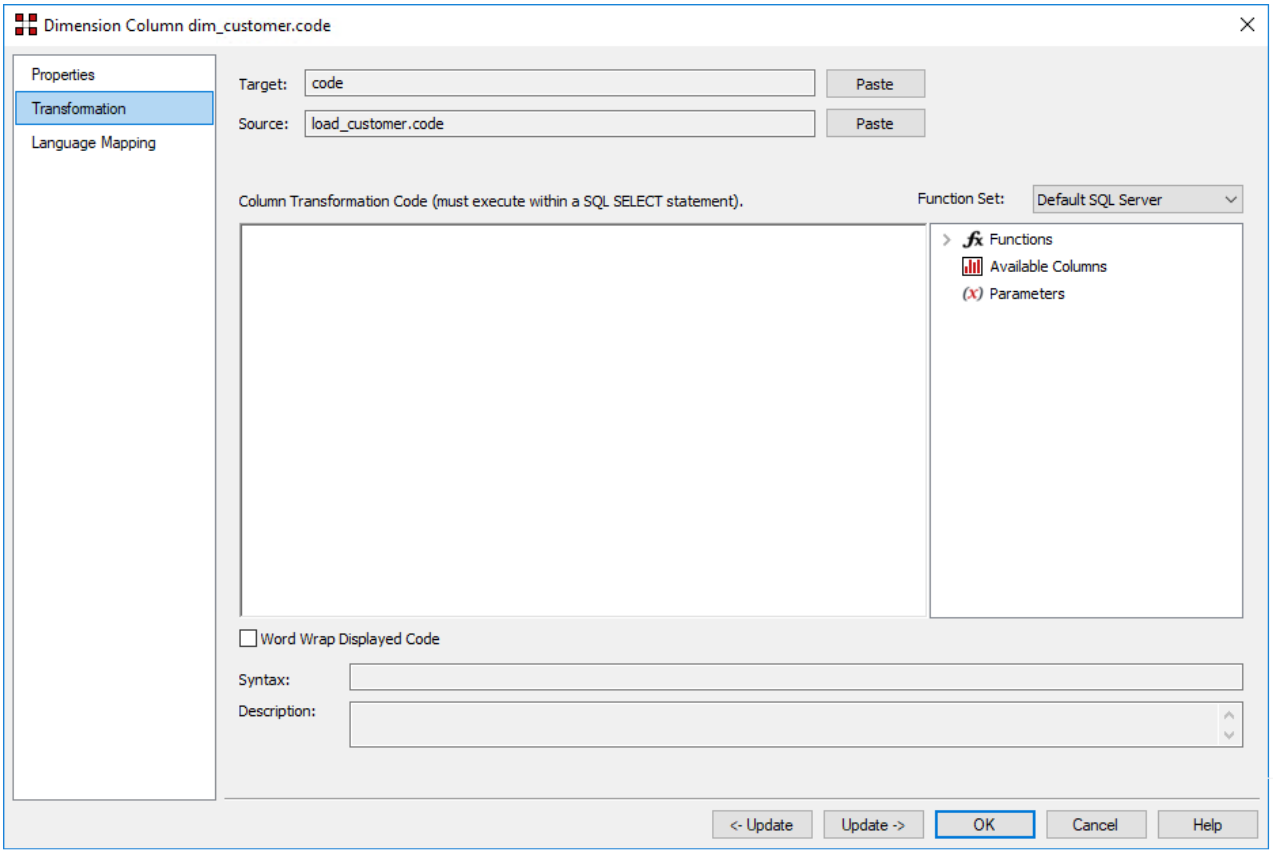
Dimension Column Transformations

Each Dimension table column can have a transformation associated with it. The transformation is included in the generated procedure and is executed as part of the procedure update.

The transformation must therefore be a valid SQL construct that can be included in a **Select** statement. For example, we could have a transformation of 'load_order_line.quantity * 0.125' to calculate a tax column of 12.5%.

Click the **Transformation** tab to enter a transformation code.

The transformation screen is as follows:



Notes:

Transformations only take effect when the procedure is re-generated.

Microsoft Analysis Services 2005+ Tabular Mode Tables: For Tabular Mode table column transformations, **Default DAX** is the only applicable Function Set for **after load** transformations.

Refer to **Transformations** for details.

Dimension Hierarchies

The various hierarchies associated with a dimension can be recorded in the WhereScape RED metadata. These hierarchies are often not used in any form, except to provide documentary completeness.

When used in conjunction with the hierarchy maintenance utility, these dimension hierarchies add a powerful method of enriching the analysis capabilities of the data warehouse. For example, we may have a source system that has a dimension called sales_person. This dimension has no information apart from employee_code, name and title. We could add additional columns of sales_manager, territory, state and region to this dimension. A hierarchy could then be formed from the salesperson name, sales_manager, territory, state and region. The hierarchy maintenance utility allows the maintenance of this hierarchy externally to the data warehouse. This external hierarchy can then become a source system to enrich the data in the warehouse.

© Copyright, WhereScape Software Ltd All rights reserved.

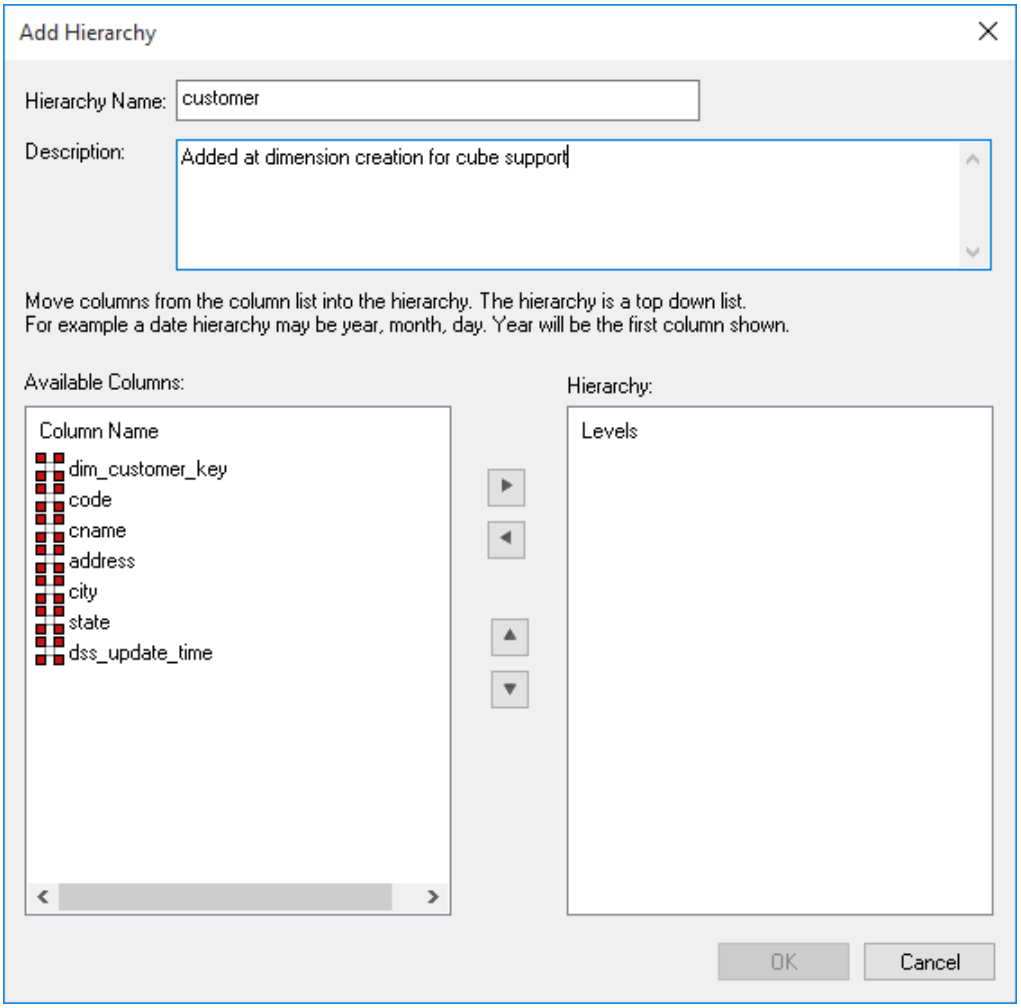
Two areas will be covered. Firstly the creation of hierarchies using WhereScape RED, and secondly the process required to setup and use externally maintained hierarchies as source systems to the data warehouse.

Adding a Dimension Hierarchy

Any number of hierarchies can be created against a dimension. There is no restriction on the form of the hierarchy.

1. To add a new hierarchy, right-click the dimension table in the left pane and select **Hierarchies > Add Hierarchy**.

   The following window appears.



2. Enter a meaningful name for the hierarchy.
3. Enter a meaningful description for the hierarchy. This description is carried through into the Hierarchy Description field of any OLAP Dimensions that are built from the original Dimension object.

Note

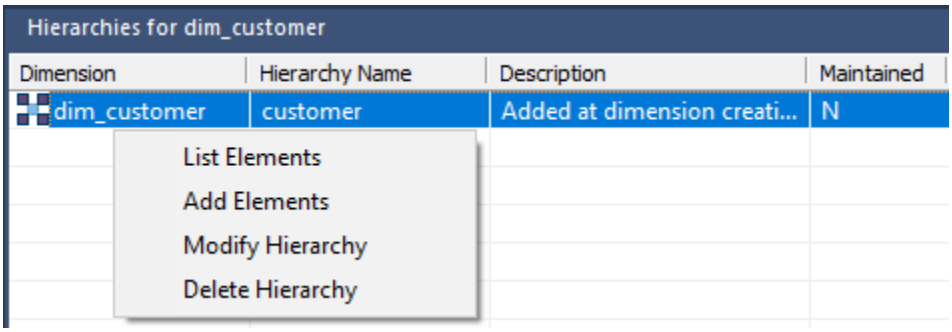
The description text is automatically set to "Added at dimension creation for cube support" but this can be edited to match the user's intended description.

The hierarchy is built with the highest level at the top; for example a customer dimension may have **category_code** at the highest level, then **group_code**, then **subgroup_code** and finally **name** at the lowest level.

1. To enter the hierarchy elements, select them, in the required order, from the left pane and click the right arrow (**>**) to add them to the right pane.

© Copyright, WhereScape Software Ltd All rights reserved.

2. Once all the hierarchy elements have been added, click **OK**.
3. A hierarchy and its elements can be edited by listing the hierarchies associated with a dimension and using the right-click menu options available in the middle pane.



Copying Dimension Hierarchies from Source

Hierarchies are automatically copied from a source table when the source table is dragged into the middle pane to create a new Dimension.

To copy the hierarchies from the source objects manually, right-click on a Dimension in the Object pane and select **Hierarchies > Copy Hierarchies from Source**. This feature is useful when the source for a Dimension has been updated to contain new hierarchies.

Snowflake

Snowflake schemas normalize dimensions to eliminate redundancy; that is, the dimension data has been grouped into multiple tables instead of one large table.

For example, a product dimension table in a star schema might be EDW 3NF into a products table, a product_category table, and a product_manufacturer table in a snowflake schema.

While this saves space, it increases the number of dimension tables and requires more foreign key joins.

The result is more complex queries and reduced query performance.

Creating a Snowflake

A snowflake dimensional structure is supported by WhereScape RED.

A snowflake can be created for EDW 3NF or partially EDW 3NF dimension tables, by including the surrogate key of the parent dimension in the child dimension.

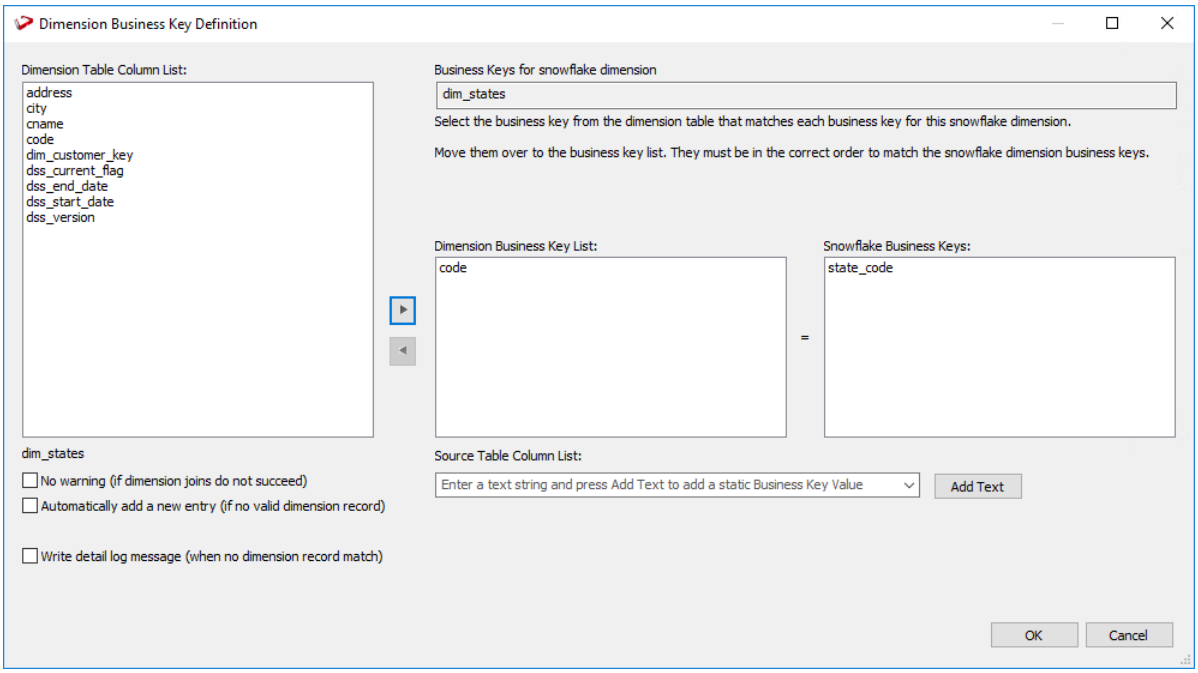
In the example below, the dim_state table represents the parent dimension.

The column dim_state_key is added to the child dimension dim_customer. Any fact tables that include the dim_customer dimension will inherently have a link to the dim_state dimension.

The process for creating a snowflake is as follows:

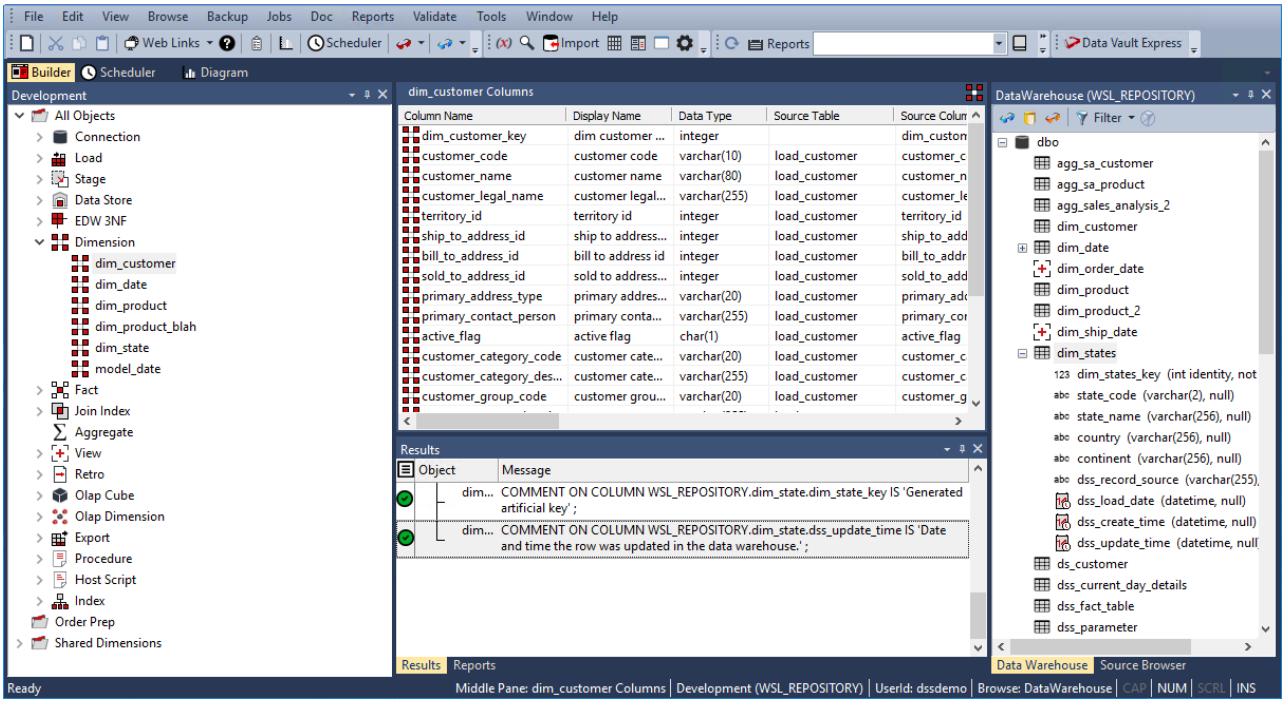
1. Build both dimensions (see previous sections).
2. Expand **dimensions** in the left pane.
3. Click on the child dimension table in the left pane to display its columns in the middle pane.
4. Browse the data warehouse connection in the right pane.
5. Expand the parent dimension table in the right pane.
6. Drag the surrogate key of the parent dimension table from the right pane to the child dimension's column list in the middle pane.
7. **Create/Recreate** the child dimension.
8. Rebuild the child dimension's update procedure.
9. A screen appears asking for the business key column(s) in the child dimension that matches the business key for the parent dimension:

© Copyright, WhereScape Software Ltd All rights reserved.



10. Add the business key column(s) and then click **OK**.

The WhereScape RED screen should look like this:

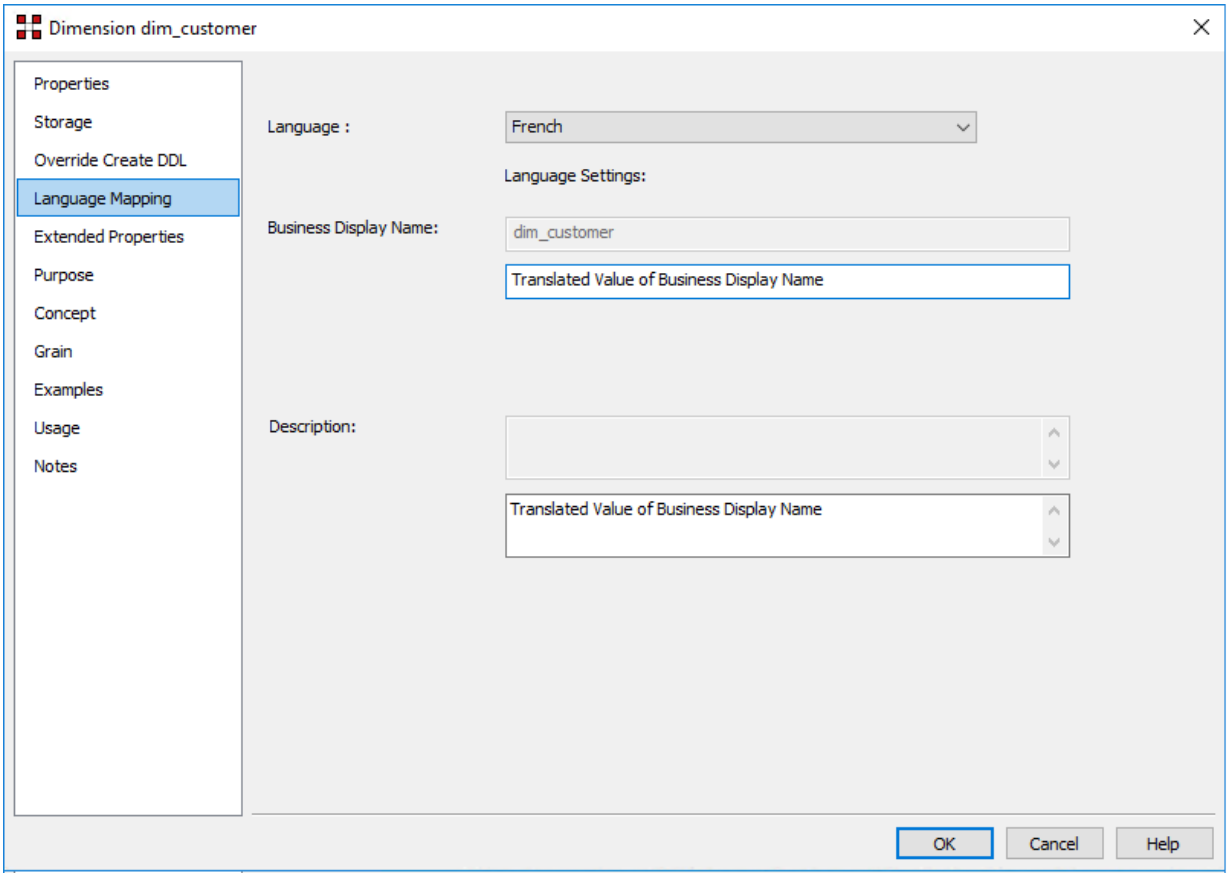


© Copyright, WhereScape Software Ltd All rights reserved.

Dimension Language Mapping

The Dimension **Properties** screen has a **Language Mapping** tab.

1. Select the language from the drop-down list.
2. Enter the translations for the **Business Display Name** and the **Description** in the chosen language.
3. The translations for these fields can then be pushed through into OLAP cubes.



To add a language to this list, refer to **Language Options** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

Staging

Stage tables are used to transform the data to a star schema or third normal form model. A stage table can be a fact or an EDW 3NF table that only contains change data or a work table. In star schema data warehouses, the stage table brings all the dimensional joins together in preparation for publishing into the fact table.

A stage table is built from the Data Warehouse connection. Unless you are retrofitting an existing system, stage tables are typically built from one or more load or stage tables. They can utilize the surrogate keys from a number of dimension tables.

The normal steps for creating a stage table are defined below and are covered in this chapter. The steps are:

1. Identify the source transactional data that will ultimately constitute the fact or EDW 3NF table. If the data is sourced from multiple tables ascertain if a join between the source tables is possible, or if a series of passes will be required to populate the stage table. If the latter option is chosen, then bespoke code is needed.
2. Using the 'drag and drop' functionality drag the table with the lowest granular data into a stage target. Refer to **Building the Stage Table** for details.
3. Add columns from other source tables. Refer to **Building the Stage Table** for details.
4. Add any relevant dimension table or EDW 3NF table keys. Refer to **Building the Stage Table** for details.
5. Create the stage table in the database. Refer to **Building the Stage Table** for details.
6. Build the update procedure. Refer to **Generating the Staging Update Procedure** for details.
7. Test the update procedure and analyze the results. Refer to **Tuning the Staging Update Process** for details.
8. Modify the update procedure as required. Refer to **Tuning the Staging Update Process** for details.

Note

If you are building a Data Vault system, a Stage table with sub type of Data Vault Stage can be created to generate hash keys that are used in building Data Vault objects (Hub, Link or Satellite tables). Refer to **Data Vaults** for details.

Building the Stage Table

Building the stage table is potentially the most challenging part of the overall task of building a data warehouse analysis area.

Most of the effort required is in the design phase, in terms of knowing what data needs to come into the fact table that will ultimately be built.

This section assumes that the decision as to what to include has been made.

Multiple Data Sources

A Stage table typically contains the change data for a detail fact table. As such, it normally maps to a specific function within the business and in many cases relates back to one main OLTP table. In many cases however, it may be necessary to combine information from a number of tables. One of the decisions required is whether or not it is practical or even possible to join the data from the different source tables.

We could however, also include two additional source tables being invoice_header and invoice_line which contain specific information relating to what was on the invoice. We may want our fact table to contain information from these tables as well. Although these two tables may contain the order_number and hence potentially allow a join with the order tables we may choose not to perform such a join for performance reasons. In this case we have three obvious choices in terms of how we ultimately update our fact table.

The choices are:

1. Join all four tables using one large join in our staging table.
2. Update the staging table in two passes; one pass updating the order information, the other pass updating the invoice information.
3. Generate two Stage tables, one for order and one for invoice. Use these two staging tables to update the one sales_detail fact table.

© Copyright, WhereScape Software Ltd All rights reserved.

Although all three options are viable and a normal situation in the WhereScape RED environment, options (2) and (3) will require specific coding and modifications to the generated procedures from the outset.

Given the example provided, option (2) would be the normal approach; although in some cases option (3) would be valid.

Drag and Drop

The best approach in creating a Stage table is to choose the source table that contains the most fields that we will be using and drag this table into the stage target.

We can then drag specific columns from the other source tables until we have all the source data that is required.

The process for defining the metadata is as follows:

1. Double-click on the **Stage Table** object group in the left pane. This will result in all existing stage tables being displayed in the middle pane. This also sets the middle pane as a **stage drop target**.
2. Browse the Data warehouse connection to display your Load tables in the right pane. This is done via the **Browse > Data Warehouse** menu option.
3. Drag the **primary load table** (i.e. the one with the most columns, or the lowest data granularity) from the right pane and drop it into the middle pane. A dialog appears to create the new staging object.
4. Leave the object type as 'Stage Table' and change the name to reflect what is being done.
5. Once a valid name is entered, the properties for the new Stage table are displayed. Normally these would be left unchanged except perhaps for storage settings.
6. Once the **Properties** window is closed, the columns for the new Stage table are displayed in the middle pane. This middle pane is now considered a **drop target for this specific stage table**. Any additional columns or tables dropped into the middle pane are considered additions to this stage table definition. Any columns that are not required can be deleted.
7. Drag and drop **additional columns** from other source tables if appropriate.
8. Drag in the **dimension artificial key** from each Dimension that is to be joined to the Stage/Fact table.
9. You can only join a Dimension, if a business key exists amongst the stage table columns or if it is possible to derive that business key in some way from the columns or other dimensions.

Note

If a column is being used to join information from two or more source tables, that column must only appear once in the Stage table. It is irrelevant which table is used to create the column in the new Stage table.

Once completed, your list of columns for the Stage table should look something like the list below. Notice the **Source Table** for each column.

stage_sales_detail Columns

Column Name	Display Name	Data Type	Source Table	Source Column
order_number	order number	numeric(12)	load_order_line	order_number
order_line_no	order line no	numeric(4)	load_order_line	order_line_no
product_code	product code	numeric(6)	load_order_line	product_code
unit_sale_price	unit sale price	numeric(9,3)	load_order_line	unit_sale_price
quantity	quantity	numeric(8)	load_order_line	quantity
sales_value	sales value	numeric(13,2)	load_order_line	sales_value
tax	tax	numeric(9,2)	load_order_line	tax
order_date	order date	datetime	load_order_header	order_date
customer_code	customer code	numeric(6)	load_order_header	customer_code
ship_date	ship date	datetime	load_order_header	ship_date
dim_customer_key	dim customer ...	integer	dim_customer	dim_customer_key
dim_order_date_key	dim time key	integer	dim_order_date	dim_order_date_key
dim_product_key	dim product key	integer	dim_product	dim_product_key
dim_ship_date_key	dim time key	integer	dim_ship_date	dim_ship_date_key
dss_update_time	dss update time	datetime		dss_update_time

© Copyright, WhereScape Software Ltd All rights reserved.

The source table (src table) reflects where each column was dragged from.

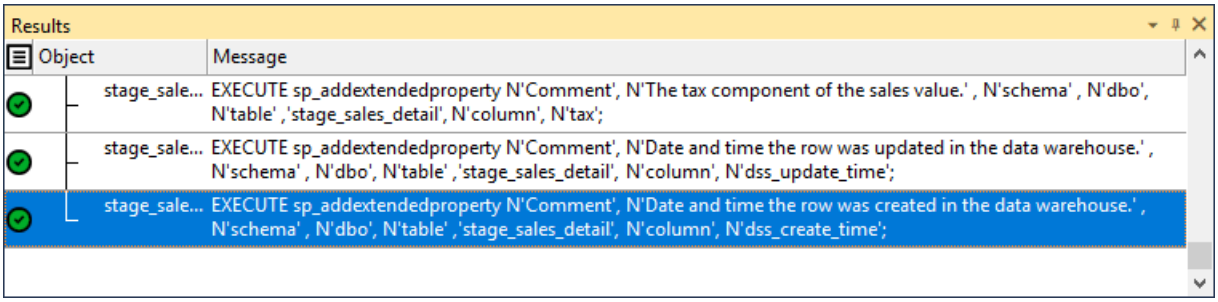
In the example above, the bulk of the columns came from the load_order_line table, and the customer_code, order_date and ship_date came from the load_order_header columns.
These two Load tables will be joined via the order_number column. This order_number column appears in both load tables but is sourced, in this example, from the load_order_line table.

Each dimension artificial key was dragged from its appropriate table. The final column 'dss_update_time' was generated by WhereScape RED and has no source.

Creating the Table

Once the Stage table has been defined in the metadata, you need to physically create the table in the database.

1. Right-click the Stage table from the left pane and select **Create (ReCreate)** from the pop-up menu.
2. The results pane displays the results of the creation. The following example shows a successful creation.



The contents of this pane is a message to the effect that the table was created followed by a copy of the actual database create statement, and if defined the results of any index creates. For the initial create no indexes will be defined.

If the table was not created then ascertain and fix the problem. A common problem is a 'Duplicate column' where a column has been accidentally added twice.

The best way to find such a column is to double-click the list heading **Column Name**, to sort the column names into alphabetical order.

Another double-click on the heading sorts the columns back into their create order. Column ordering can be changed by altering the column order value against a column's Properties.

Tip

Double-clicking the heading of a column in a list, sorts the list into alphabetical order based on the column chosen.

Generating the Staging Update Procedure

Once a stage table has been defined in the metadata and created in the database, an update procedure can be generated to handle the joining of any tables and the lookup of the dimension table artificial keys.

Note

You can also generate an update procedure via a template, refer to **Rebuilding Update Procedures** for details.

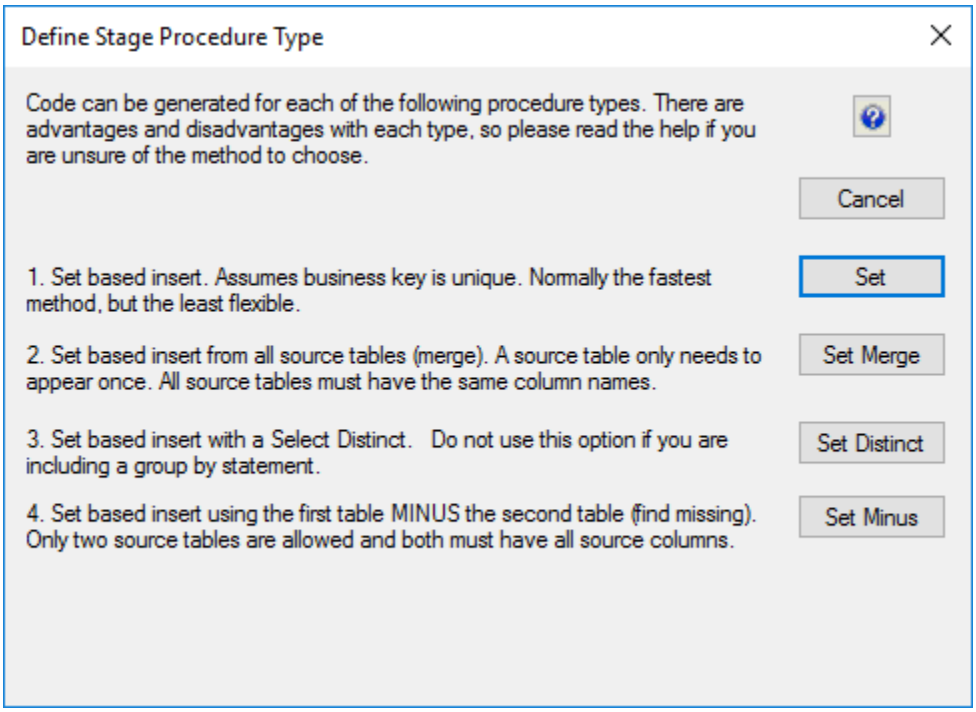
Generating a Procedure

1. To generate a procedure, right-click on the stage table in the left pane and select **Properties**.
2. From the **Update Procedure** drop-down list, select **(Build Procedure...)**.
3. Click **OK** to update the properties and start the process of generating the new procedure.
4. A series of prompts are displayed during the procedure generation to join the tables and link the dimensions.

© Copyright, WhereScape Software Ltd All rights reserved.

Procedure Type

The first dialog box asks for the type of procedure that is to be generated. An example of this follows:



Note

Default settings for hint and truncate options are set in **Tools > Options > Code Generation > Default Update Procedure Options**.

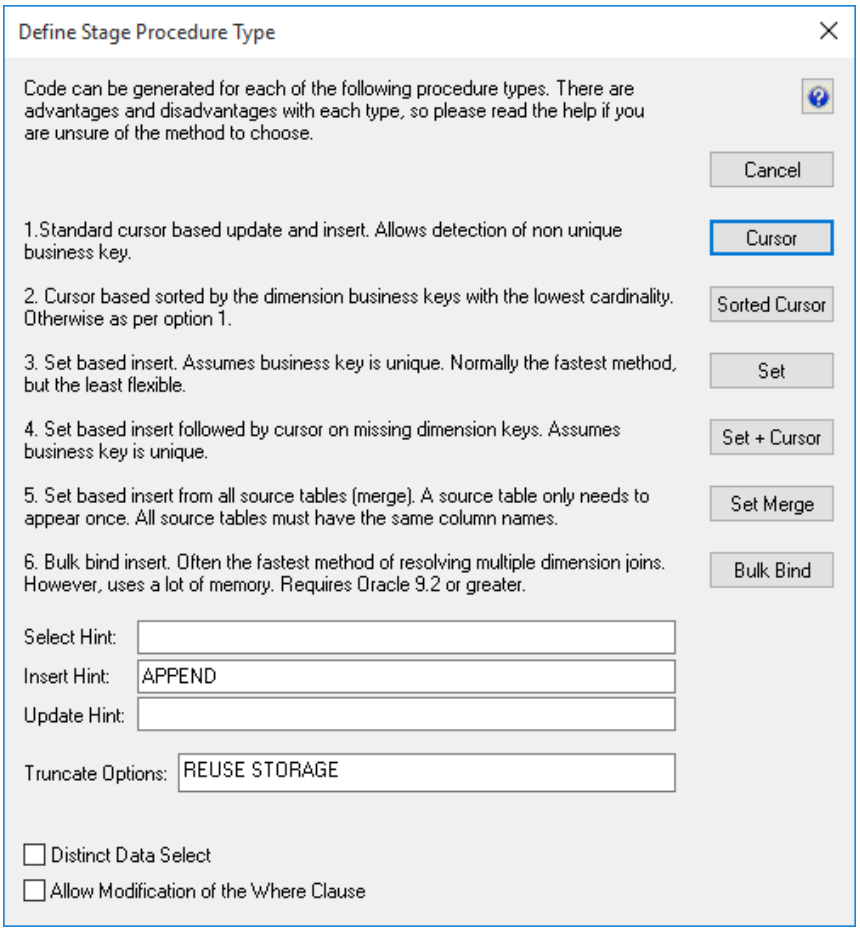
© Copyright, WhereScape Software Ltd All rights reserved.

Oracle

Tip

Oracle Only

- Select Hint: Enter a database hint to be used in the SELECT statement.
- Insert Hint: Enter a database hint to be used in the INSERT statement. Factory default value is APPEND.
- Update Hint: Enter a database hint to be used in the UPDATE statement.
- Truncate Options: Optional Oracle truncate options such as **REUSE STORAGE** (default value) or **DROP STORAGE**.



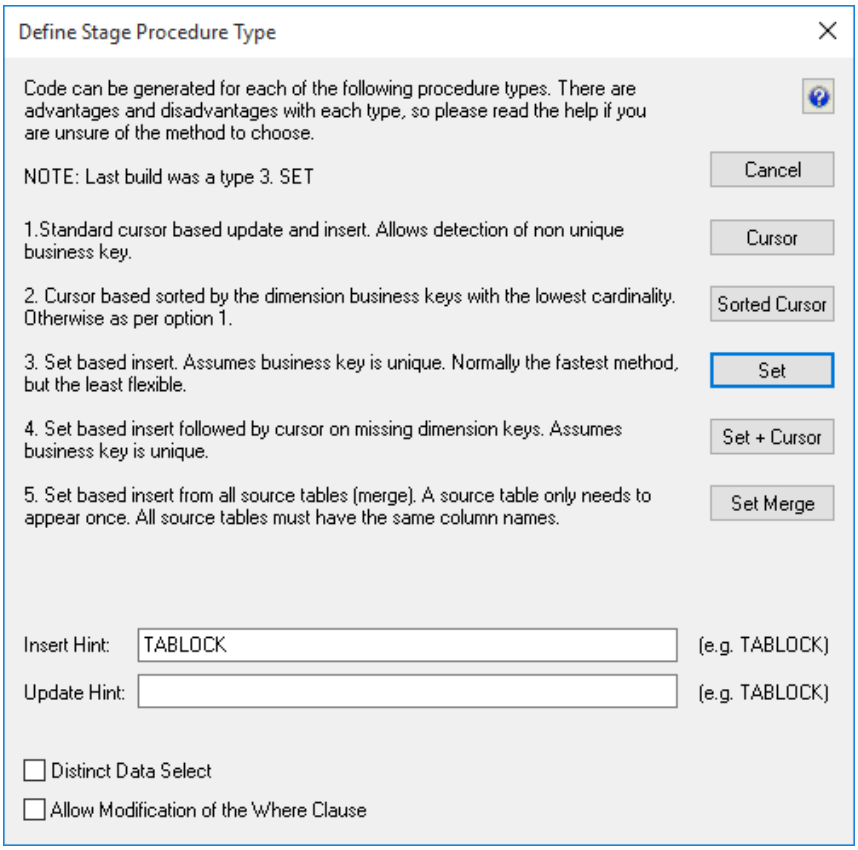
© Copyright, WhereScape Software Ltd All rights reserved.

SQL Server

Tip

SQL Server Only

- Enter a database hint to be used in the INSERT statement. Factory default value is TABLOCK.
- Enter a database hint to be used in the UPDATE statement.



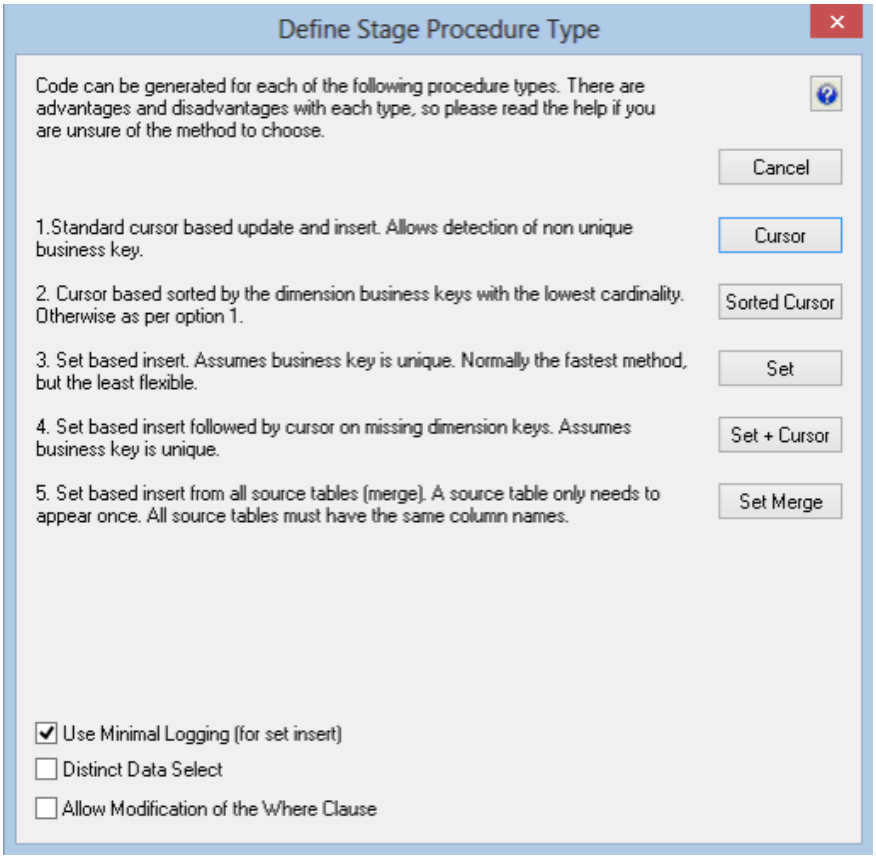
© Copyright, WhereScape Software Ltd All rights reserved.

DB2

Tip

DB2 Only

- Use Minimal Logging (for set insert): The staging procedure uses the **LOAD FROM CURSOR** option in DB2 to minimize database logging.



A number of different types of procedure generation are available. Each type is discussed below.

There is a check-box at the bottom of the dialog box. This check-box is only present if advanced procedure building features are enabled.

This check-box allows the editing of the 'Where' clause when no table joining is being performed, and hence the 'Where' clause would not be exposed.

Cursor Based Procedure

This option creates a procedure that loops through each source transaction performing specific tasks on each transaction before writing the transaction to the stage table. The tasks performed include the lookup of each dimension key, and any bespoke value assignments or adjustments.

Sorted Cursor Procedure

This is a variant on the regular cursor procedure. In this case dimension key lookups only occur when there is a change in the business key for a given dimension. This can improve the performance of the procedure particularly when one or more of the dimensions has a low cardinality. Where slowly changing dimensions are used this advantage is lessened as the relative date of the dimension records also comes into play.

Set based procedure

A set based procedure performs one SQL statement to join all the dimensions and source tables together and then insert this data into the stage table. This is normally the fastest method of building the stage table. It is also the least flexible method and is easily prone to incorrect syntax. Caution and experience must be used with this option to ensure that all source transactions are processed and that they are processed correctly. This option is only recommended for the experienced practitioner where performance is the overriding

© Copyright, WhereScape Software Ltd All rights reserved.

requirement. Particular care must be taken with regards to **NULL** values in the business keys that are used to make the table joins. The generated code deliberately does not handle such null values.

Set + Cursor Procedure

This option is normally used where a number of dimension keys will fail to join and we wish to automatically add new dimension records. The first phase of the generated code will perform a set based join and insert of the data. The dimensions that failed to join will be assigned the zero key value. The second phase of the procedure will use a cursor to cycle through all the stage records that have a zero key dimension join. By setting the appropriate flags in the dimension joins when building this procedure we can request that any missing dimension records be automatically added.

Set Merge Procedure

This option is to allow the merging of two or more identical tables. The tables to be merged must have exactly the same number of columns and column names. If necessary additional blank columns could be added to one or other of the tables to make them identical. To use this procedure you must simply have the tables to be merged mentioned at least once in the **Source Table** field of a column's properties. Refer to **Set Merge Procedure** for details.

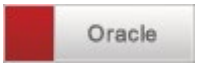
Set Distinct

Essentially the same as Set, except for the DISTINCT key word being added to the SELECT statement. This option therefore removes duplicate rows.

Set Minus

The Set Minus option can be used to determine change data or for programmatic referential integrity checking. This option works in a similar way to Set Merge. It generates SQL code in this form: *SELECT ... FROM source_table1 {where} MINUS SELECT ... FROM source_table2 {where}*. It requires exactly two source tables to be specified. All source columns must exist in both source tables.

Bulk Bind Procedure



The bulk bind procedure is only available for Oracle. It uses memory arrays to join all the information before writing it to the table. It can be the fastest method of performing a stage table update when a lot of dimension joins are being used.

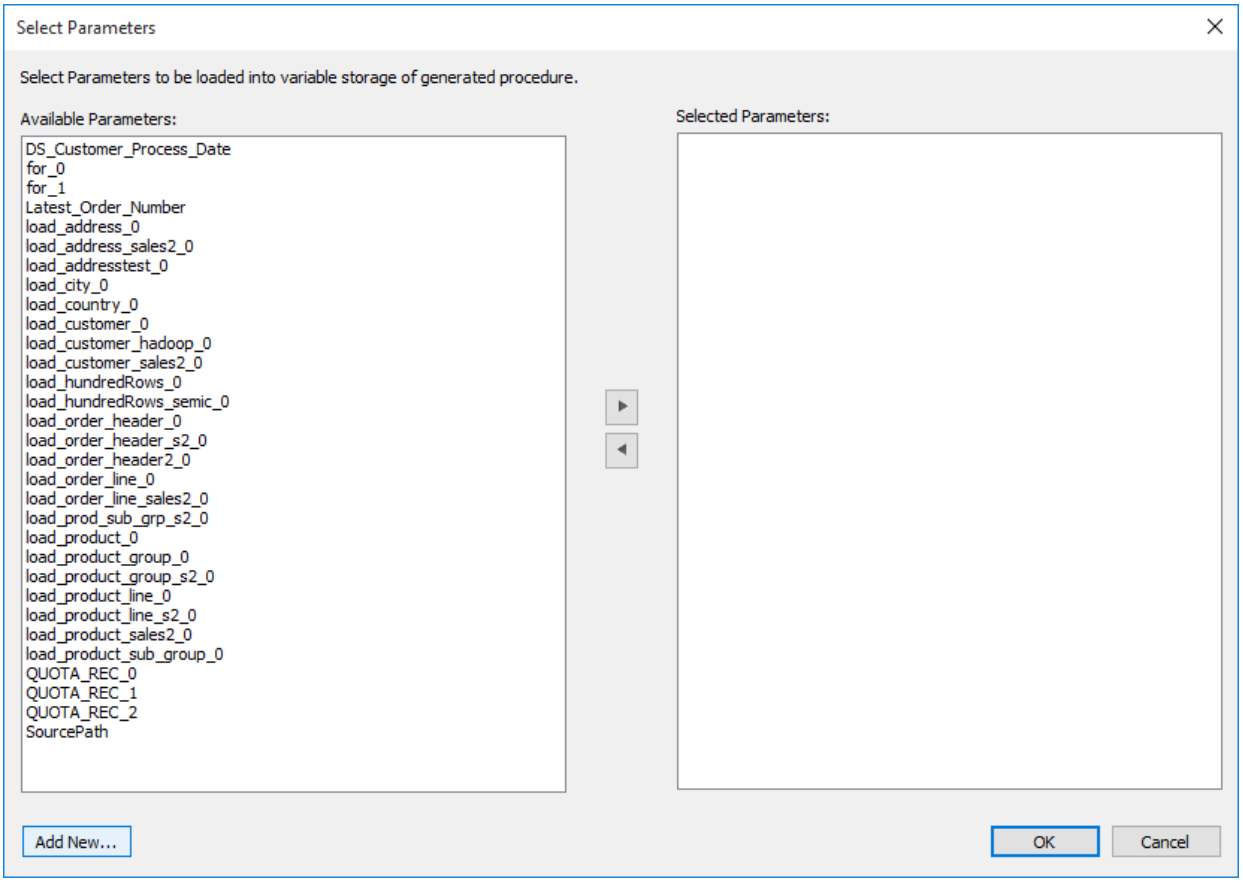
There are some other additional options available in this dialog common to SQL Server, Oracle and DB2:

- **Allow modification of the Where clause:** the 'Where' clause dialog is displayed automatically if the table has more than one source table, but not for a single source table stage table unless this option is chosen.
- **Distinct data select:** choosing this option adds the DISTINCT key word to the SELECT statement in the staging procedure.

© Copyright, WhereScape Software Ltd All rights reserved.

Parameters

If WhereScape RED parameters exist in the metadata, the following window is displayed. Any parameters selected in this dialog (by moving them to the right pane), are included in the generated update procedure as variables.



The variables can also be used in column transformations and in the from/where clause for the update procedure.

Some databases have a 30 character limit for variable names. WhereScape RED ensures the variables added for any parameters are less than 30 characters long by creating variable names in the form v_ followed by the first 28 characters of the parameter name.

For example, a parameter called MINIMUM_ORDER_NUMBER_SINCE_LAST_SOURCE_LOAD will be available as the variable v_MINIMUM_ORDER_NUMBER_SINCE_L.

Tips

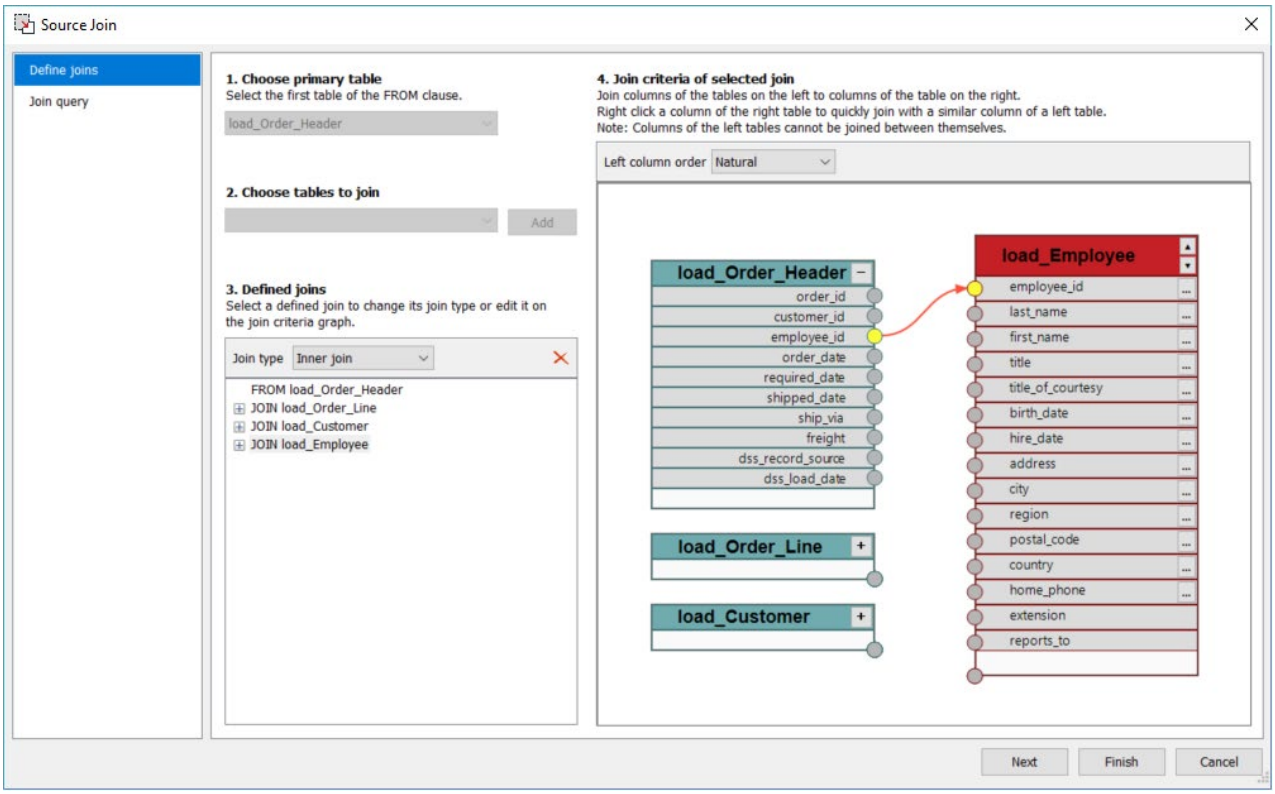
- WhereScape RED parameters should be unique within the first 28 characters to avoid conflicting variables names.
- If the desired parameter doesn't exist in the metadata yet, a new parameter can be added by clicking on the **Add New** button on the bottom leftmost corner of the Select Parameters dialog.

Refer to **Parameters** for more information on WhereScape RED Parameters.

© Copyright, WhereScape Software Ltd All rights reserved.

Multiple Source Tables

WhereScape RED provides a wizard that enables you to graphically generate the join clause for an object with multiple source tables. The **Source Join** wizard enables you to specify the primary (driving) table and the source tables to join. A graphical representation of the tables and the join criteria is displayed by the wizard which you can manipulate to change the defined joins.



The **Source Join** wizard guides you in generating the join clause through the following steps:

1. **Choose Primary Table**
   The first source table encountered is the default primary (driving) table which can be changed before any joins are defined.
2. **Choose Tables to Join**
   Once the primary table is set, the remaining source tables are listed in this drop down. Select a table to join and then click **Add**. The two tables are graphically added in pane 4, which is the basis for the join displayed in pane 3.
3. **Defined Joins**
   The join type defaults to **Inner** join but this can be changed from the drop down.
4. **Join criteria of selected join**
   The column(s) used to join the two tables can be joined manually by clicking the corresponding join column connection point from the table on the left and then dragging the line that appears to the connection point of the required column in the table on the right.
   Alternatively, if a column with a matching or similar name exists in the left hand table(s) then a context menu is provided on the columns on the right-hand table. You can right click on the column to display the context menu and then select a column from the list.

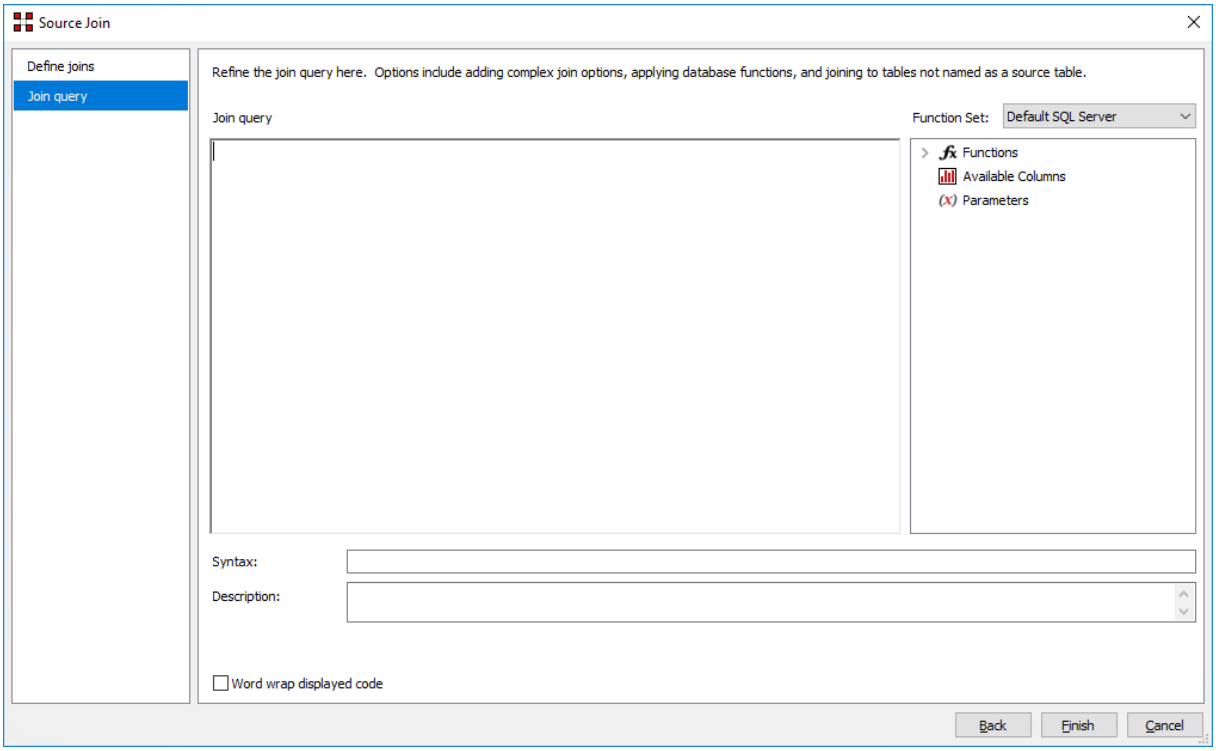
   Tip

   To aid manual column join, you can expand or collapse columns in the left-hand tables and toggle between natural and alphabetic column ordering.
5. Once the join column(s) for the table pair are defined, then any additional source tables can be added from the drop-down list in step 2 and the above column mapping process repeat to join this table.

   This process is repeated until all source tables are joined.

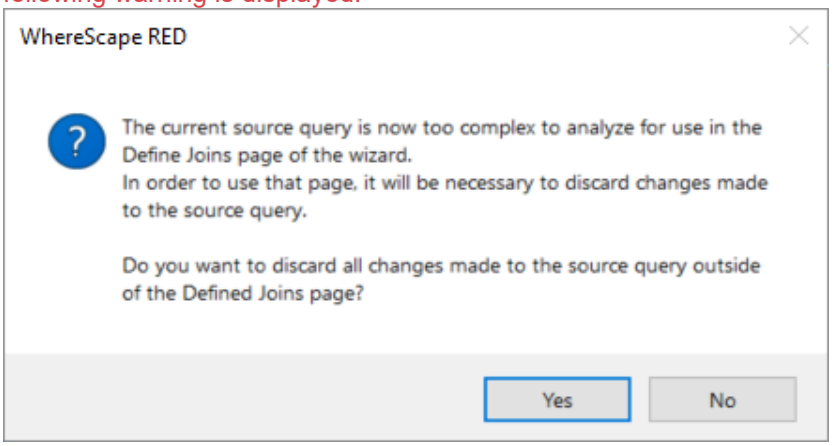
© Copyright, WhereScape Software Ltd All rights reserved.

6. Clicking the **Next** button or the **Join Query** tab progresses to the join editor window, where custom joins, database functions, etc. can be added.



Note

If RED is unable to parse a join statement, the user cannot navigate back to the define join tab and the following warning is displayed:



Dimension Joins

For each dimension key, a dialog appears asking for the business key from the stage table that matches the business key for the dimension.

In the example below, we are asked for the stage table business key for the customer dimension. The dimension name is shown both on the first prompt line and at the lower left side of the dialog box.

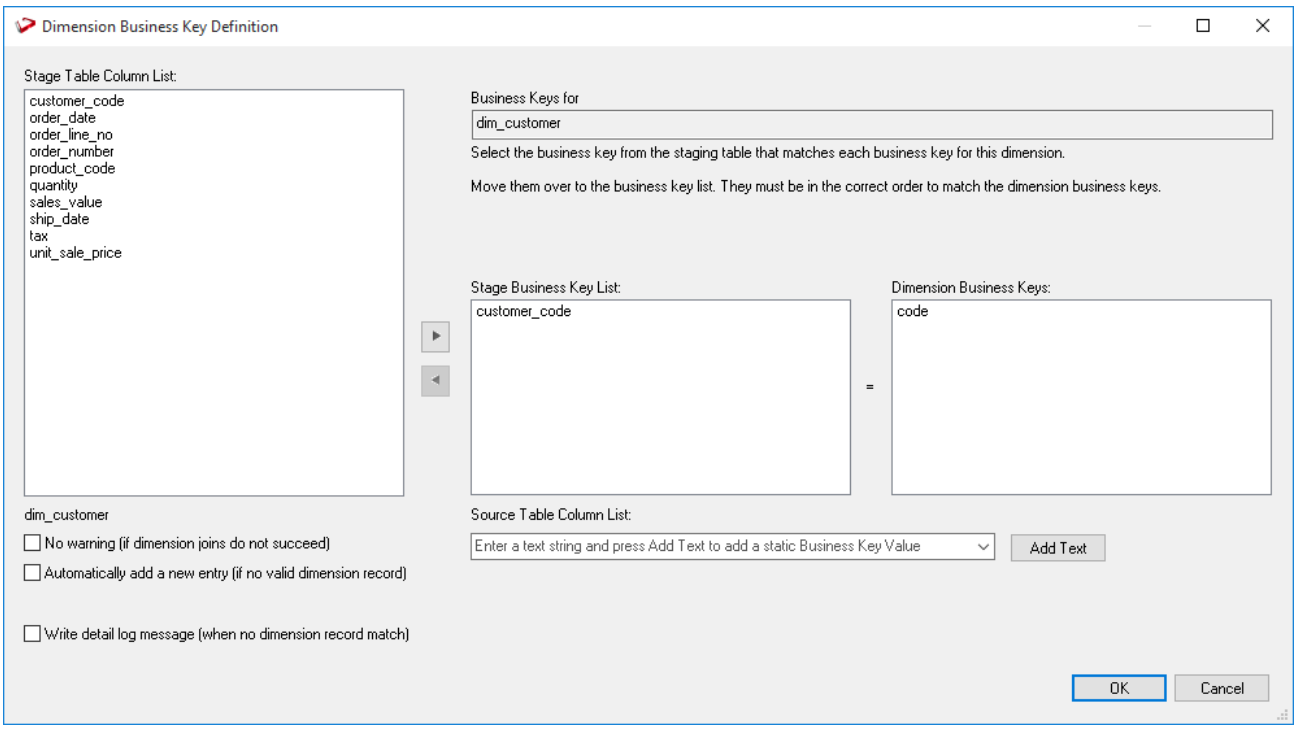
The customer dimension has a unique business key named **code**.

We must provide the corresponding business key from the staging table. In the case of our example, this is the customer_code column.

1. Click **OK** after the correct business key has been entered.

© Copyright, WhereScape Software Ltd All rights reserved.

2. If the business key does not exist and will be derived from another dimension or from some form of lookup then enter any column and edit the procedure once produced.



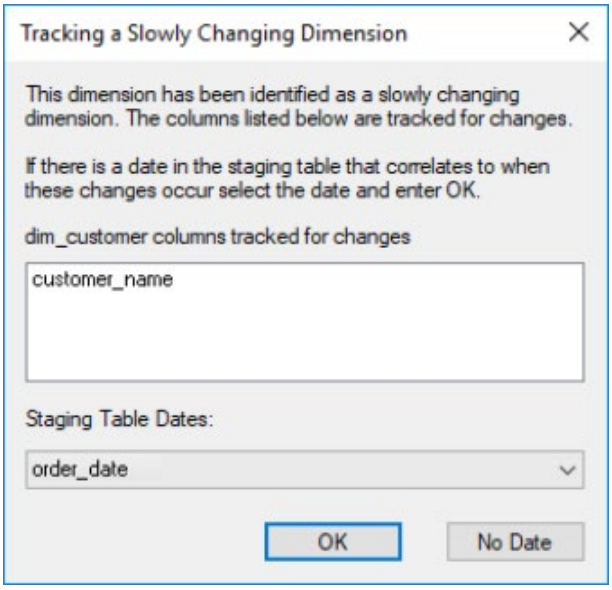
Note

The **Add Text** button and the associated message and edit box are only shown if the user possesses a full license, thus enabling the advanced procedure build options. When the **Add Text** button is clicked, any data in the edit box to the right of the button is placed in the Stage table column list. In this way, a number or string can be assigned as part or all of a Dimension join.

Slowly Changing Dimension information

If the dimension being joined was defined as a slowly changing dimension, then an additional window appears, as shown below. This window asks for a date field in the Stage table that enables RED to determine which version of the slowly changing dimension (the **customer_name** field, below) to use based on the specified date range in the Customer Dimension.

Select the appropriate date field for your business needs and click **OK**. If you wish to take the last (or current) version for the dimension, select **No Date**.



© Copyright, WhereScape Software Ltd All rights reserved.

For Example:

As seen in the screen above, we have defined the **customer_name** as an item in the dimension that we expect to have versions for over time, e.g. each time the data warehouse processing sees a new **customer_name** value, the dimension will record the date range for that version's validity even though the business key (customer_code in this example) remains the same. This implies we want to create a new dimensional record whenever a customer name is changed even though the customer_code remains the same.

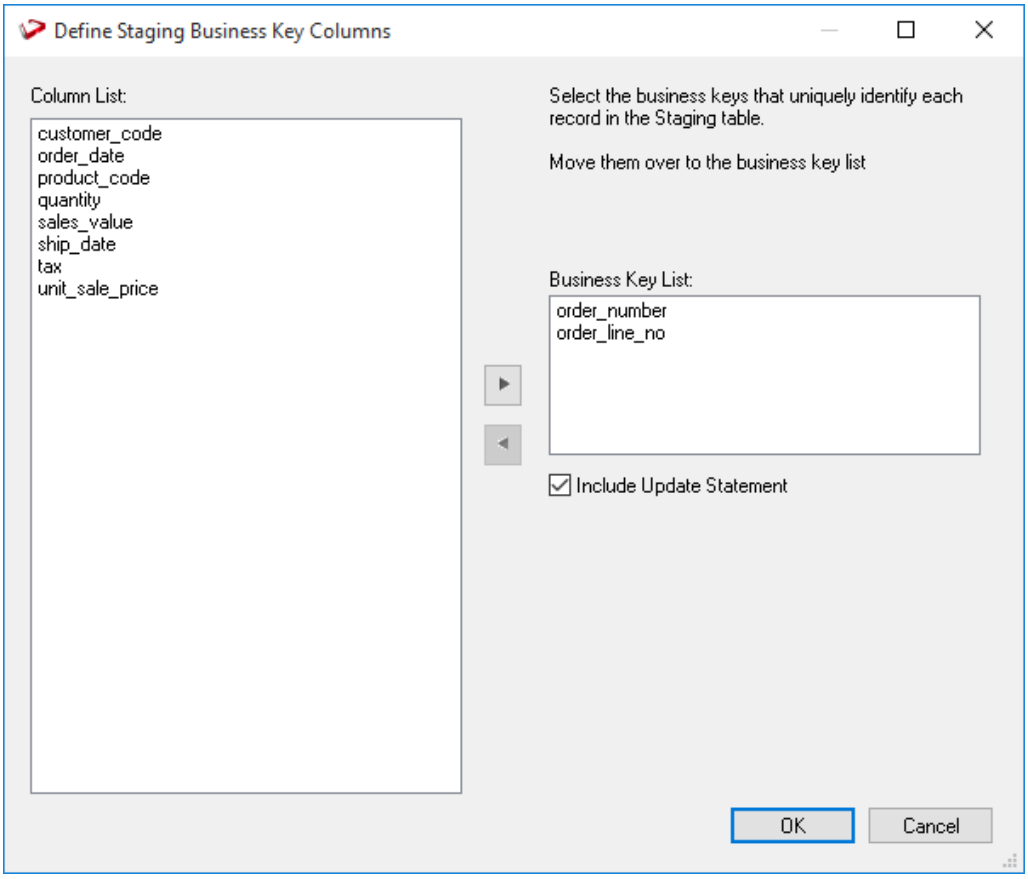
Let's say a customer changes their name on the 5th of the month. If the **Staging Table Dates** field is set to **order_date**, any order received before the 5th of the month is identified under the old customer name and any order received on or after the 5th has the new customer name.

Alternatively, by setting the **Staging Table Dates** to **ship_date**, we can specify that any order **shipped** on or after the 5th of the month is shipped with the new name.

Staging Business Key

Once all the dimensional joins have been defined, we will be asked to define the unique business key for the staging table. This is the column or columns that allow us to uniquely identify each record in the staging table.

In the example below, the unique business key is a combination of the order_number and the order_line_no.



Notes

- The order of the columns in the business key list is set to the order that the columns appear in the table. This is done to optimize performance for any indexes.
- **NULL VALUES:** None of the columns chosen as the business key should ever contain a NULL value. See the note at the start of the Dimension chapter.

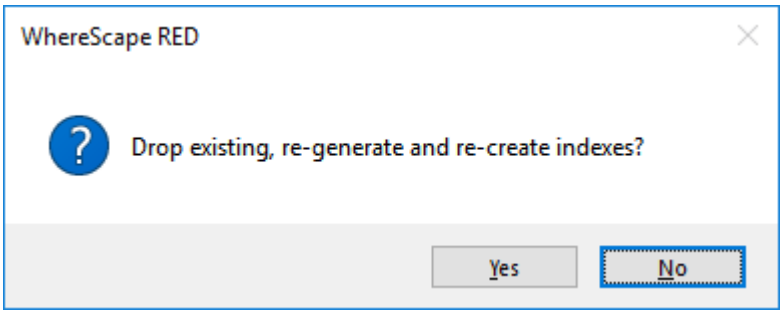
The **Include update statement** check-box provides two options in the generated procedure. If selected, then an update/insert combination will be included. If clear, then only an Insert statement will be included in the procedure. If cleared, you must be sure that the business key for the table is unique, otherwise either a duplicate key error will occur or records will be lost depending on the type of procedure.

© Copyright, WhereScape Software Ltd All rights reserved.

Index re-creation

Finally, you are asked if you want to drop and re-create any indexes defined on the table:

- Click **Yes** to drop and re-create.
- **No** to leave existing indexes in place.



Building and Compiling the Procedure

- Once the above questions are completed the procedure is built and compiled automatically.
- If the compile fails, an error is displayed along with the first few lines of error messages. Compilation failures typically occur when the physical creation of the table was not done.
- If the compile fails for some other reason, the best approach is to use the procedure editor to edit and compile the procedure. The procedure editor highlights all the errors within the context of the procedure.
- Once the procedure has been successfully compiled, it can either be executed interactively or passed to the scheduler.

Tuning the Staging Update Process

This section is for Cursor based procedures only. It is not relevant for Set based or other types.

When generating the cursor based procedures to update the staging table WhereScape RED includes Update/Insert logic. This logic first attempts to update an existing staging record and then inserts a new record if the record does not exist.

This build option should be altered once an understanding of the data you are dealing with is acquired.

For the initial prototype and testing phases of a data warehouse analysis area we need to find out if our business keys for the fact table are genuinely unique. The update/insert logic in the staging procedure will only perform an 'Update' if our business key is not unique.

In this case we need to decide if this is a failure in the business key, or some form of additive record situation we need to handle and code accordingly.

© Copyright, WhereScape Software Ltd All rights reserved.

Removing the Update Component

Once we have resolved any data issues we would normally remove the update component to speed up the processing of the stage data. There are two ways of doing this. First we can clear the Include Update Statement check-box when defining the table business key. This will result in a procedure with only an insert statement. The other option is to do it manually as follows:

There are normally three steps to removing the update component. These being:

1. The update makes use of an index that can also be deleted in most situations.
2. The actual Update statement can be removed from the code such that the insert is always executed.
3. The procedure also includes an analyze of the table after 1000 rows have been inserted. This code is present to ensure that the index is used in the update. We can now remove this code. It would normally look like the following (Oracle version):

```
--=======================================
-- If 1000 or more rows inserted then perform an analyze
--=======================================
IF v_insert_count >= 1000 And v_analyze_flag = 0 THEN
    v_analyze_flag := 1;
    v_Sql := 'analyze table stage estimate statistics sample 5 percent for all
indexed columns';
    execute immediate v_Sql;
END IF;
```

© Copyright, WhereScape Software Ltd All rights reserved.

Stage Table Column Properties

Each stage table column has a set of associated properties. The definition of each property is defined below:

If the **Column name** or **Data type** is changed for a column then the metadata will differ from the table as recorded in the database.

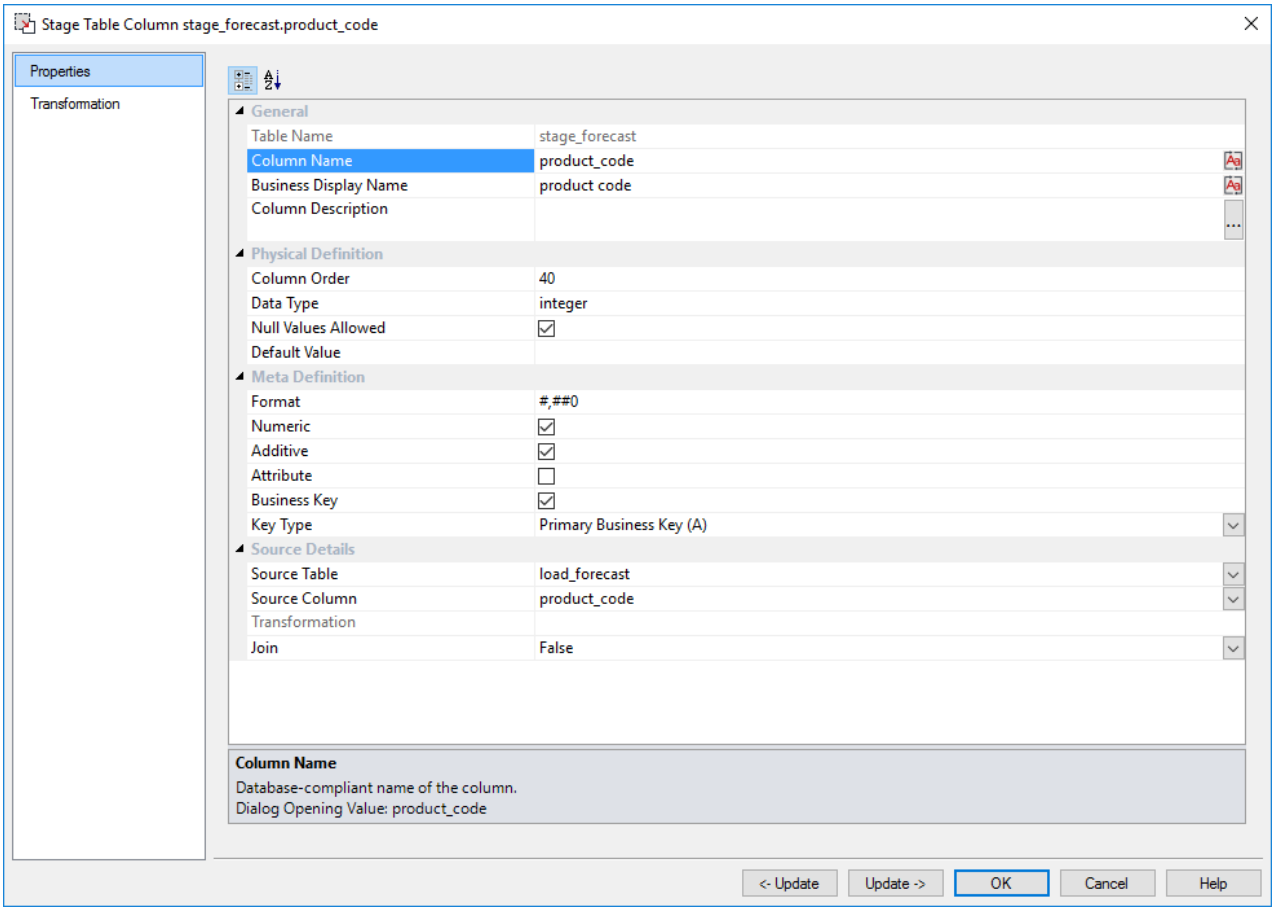
Use the **Validate>Validate Table Create Status** menu option to compare the metadata to the table in the database.

A right mouse menu option of **Alter Table** is available when positioned on the table name after the validate has completed. This option will alter the database table to match the metadata definition.

Tip

If a database table's definition is changed in the metadata then the table need to be altered in the database. Use the **Validate>Validate Table Create Status** menu to compare metadata definitions to physical database tables. The option also provides the ability to alter the database table, through a pop-up menu option from the validated table name.

A sample **Properties** screen is as follows:



Tip

The two special update keys enables you to update the column and step either forward or backward to the next column's properties.
ALT-Left Arrow and **ALT-Right Arrow** can also be used instead of the two special update keys.

General

Options	Description
Table Name	Database-compliant name of the table that contains the column. [Read-only].

© Copyright, WhereScape Software Ltd All rights reserved.

Options	Description
Column Name	Database-compliant name of the column. Typically column-naming standards exclude the use of spaces etc. A good practice is to only use alphanumerics, and the underscore character. Changing this field alters the table's definition. **Note** A case conversion button on the right converts the text between different cases: UPPER CASE, Capitalized Case and lower case. The mode cycles to the next case in the sequence each conversion.
Business Display Name	Name that the business uses to refer to the column, which is included in the RED-generated documentation and can be used in the end user layer of other tools. [Does NOT affect the physical database table]. As such it is a free form entry and any characters are valid. **Note** A case conversion button on the right converts the text between different cases: UPPER CASE, Capitalized Case and lower case. The mode cycles to the next case in the sequence each conversion.
Column Description	This field contains the description for the column. It may be a description from a business user's point of view. This field might additionally contain information on where and how the column was acquired. For example if the column is sourced from multiple tables or is a composite or derived column then this definition would normally describe the process used to populate the column. This field is used in the documentation and is available via the view **ws_admin_v_dim_col .** This field is also stored as a comment against the column in the database.

Physical Definition

Options	Description
Column Order	Numeric value that controls the relative order of columns in the database create statement. The lowest numbered column will appear first in the table. Although this affects the physical table definition no action will be taken unless the table is re-created in the database. The columns can be re-numbered based on the existing order by choosing the **Respace Order Number** pop-up menu option when positioned over any column in the table. This action will number the columns in increments of 10 starting at 10. In addition to a simple change of the order field, the column order can be changed by first displaying the columns in the middle pane and then using drag and drop to move the columns around. This drag and drop process will automatically renumber the columns as required.
Data Type	Database-compliant data type that must be valid for the target database. Typical data types for Oracle are integer, number, char(), varchar2() and date. For SQL Server, common types are integer, numeric, varchar() and datetime. See the database documentation for a description of the data types available. Changing this field alters the table's definition.
Null Values Allowed	Determines whether the table column can hold NULL values or whether a value is always mandatory.
Default Value	Initial value that is assigned to the column when a row is inserted into the table but no value is specified for the column.
Format	Optional format mask that can be used in end user tools. [Does NOT affect the physical database table]. As such it is a free form entry and any characters are valid. Typically format masks are only used on numeric fields. Example: #,###0.00. It is not worth the effort of populating this field unless it can be utilized by the end user tools in use.

© Copyright, WhereScape Software Ltd All rights reserved.

Meta Definition

Options	Description
Numeric	Indicates whether the table column holds values that are numeric. This is normally only relevant for fact tables. It does not affect the physical table definition, but rather provides input to the view **ws_admin_v_dim_col** which can be used to assist in the population of an end user tool's end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.
Additive	Indicates whether the table column holds values that are additive. This implies that the column can be summed when performing data grouping in a query. This is normally only relevant for fact tables. It does not affect the physical table definition, but rather provides input to the view **ws_admin_v_dim_col** which can be used to assist in the population of an end user tool's end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.
Attribute	Indicates whether the table column holds values that are descriptive, and/or are used for grouping/summing. An attribute is defined as a column that is non factual. For example we may have an order number, or a invoice number stored in the fact table. Such columns are considered attributes, rather than facts. This checkbox is therefore normally only relevant for fact tables. This checkbox does not affect the physical table definition, but rather provides input to the view **ws_admin_v_dim_col** which can be used to assist in the population of an end user tools end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.
Business Key	Indicates whether the column is part of the BUSINESS key, which is defined during the update procedure build. [Normally maintained automatically]. Multiple columns can form the primary business key.
Key Type	Key type that is assigned and used when generating the table's update procedure and indexes. [Normally maintained automatically]. It can be altered here, but this should only be done if the consequences are fully understood and tested. The supported values are:

Key type	Meaning
0	The artificial key. Set when the key is added during drag and drop table generation.
1	Component of all business keys. Indicates that this column is used as part of any business key. For example: By default the dss_source_system_key is added to every dimension table. It is considered part of any lookup on that table and has the key type set to 1. Set when the column is added during drag and drop table generation.
2	Indicates that this column is a dimensional join. Used on fact tables to indicate the dimension keys. Results in bitmap indexes being built for the columns. Set during the update procedure generation for a fact table, based on information from the staging table.
3	Slowly changing column indicator. Used on dimension tables to indicate that the column is being managed as a slowly changing column within the context of a slowly changing dimension. Set when a column is identified during the dimension update procedure generation.
4	Previous value column indicator. Used on dimension tables to indicate that the column is being managed as a previous value column. The source column identifies the parent column. Set during the dimension creation.
5	Indicates a column is a start date column.
6	Indicates a column is a end date column.
7	History column indicator. Used in model history tables to indicate that the column is being managed as a history column within the context of a model history table. Set when a column is identified during the model history update

© Copyright, WhereScape Software Ltd All rights reserved.

Options	Description	
		procedure generation.
	c	Change Hash Key column indicator. Used in Data Vault tables to indicate the differences in the descriptive columns of a Satellite table. Refer to **Data Vaults** for details.
	h	Hub Hash Key column indicator. Used in Data Vault tables to indicate the hash key column of a Hub Table. Refer to **Data Vaults** for details.
	l	Link Hash Key column indicator. Used in Data Vault tables to indicate the hash key column of a Link Table. Refer to **Data Vaults** for details.
	A	Indicates that the column is part of the primary business key. Set whenever a business key is defined as part of an update procedure generation.
	B-Z	Indicates that the column is part of a secondary business key. Only used during index generation and not normally set.
Hash Key Sources	This field is only displayed for Hash key types. Displays the hash source columns that are used to generate the selected Hub, Link or Change hash key.	
Hash Key Source For	This field is only displayed for Hash key types. Displays the hash keys columns that use the displayed hash key sources.	

Source Details

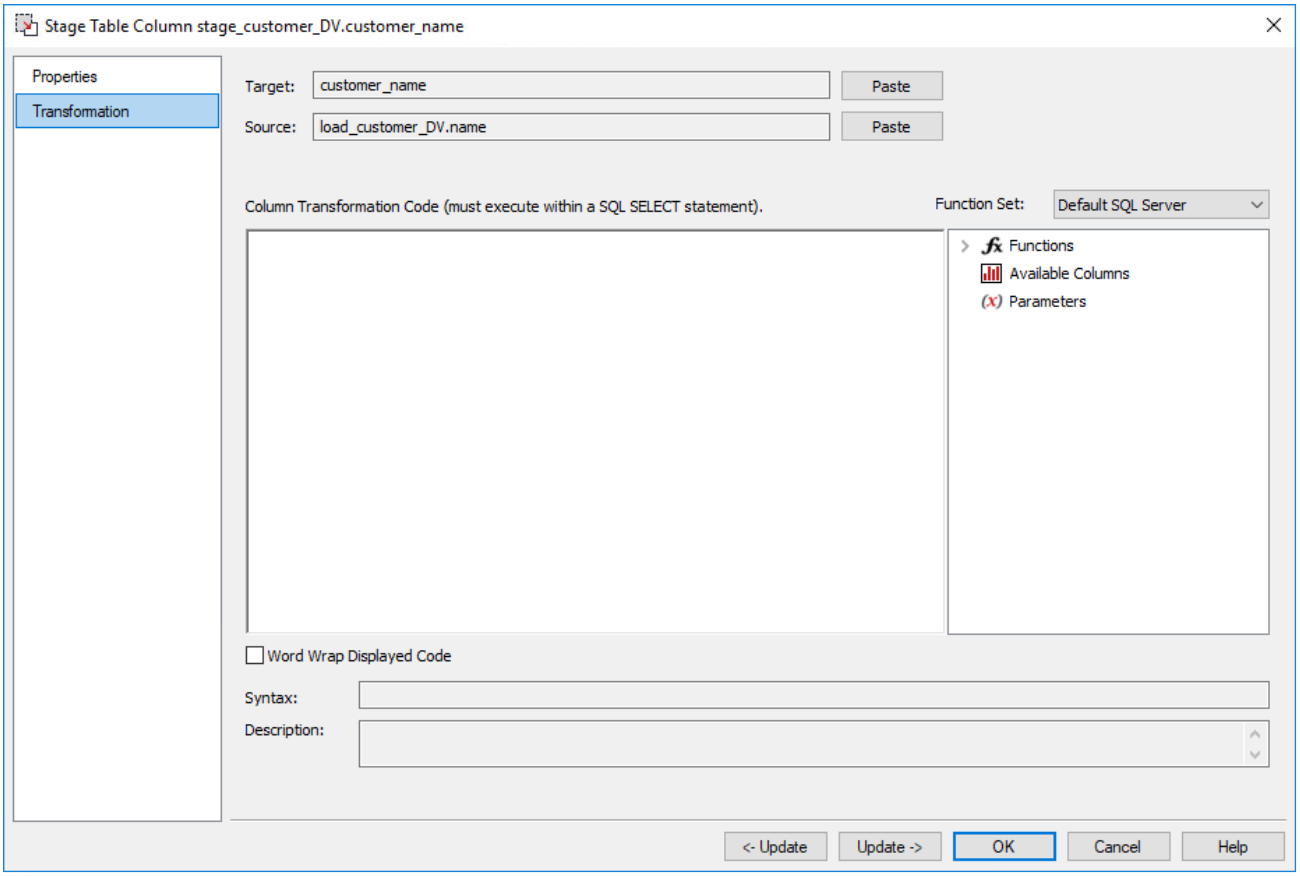
Options	Description
Source Table	Identifies the source table where the column's data comes from. This source table is normally a load table, or a dimension table within the data warehouse. If the column was sourced from multiple tables, then the normal practice is to record one of the tables in this field and a comment listing all of the other tables in the **Source strategy** field. This field is used when generating a procedure to update the stage table. It is also used in the track back diagrams and in the documentation.
Source Column	Identifies the source column where the column's data comes from. Such a column is normally a load table column, which in turn may have been a transformation or the combination of multiple columns. This may also be a dimensional key where a dimension is being joined.
Source Data Type	Identifies the source column's data type. [Read-only].
Transformation	Refer to **Stage Table Column Transformations** for details. [Read-only].
Join	Indicates whether the table column is used in a table join. [Normally maintained automatically but can be optionally changed to override the default join logic used in the generated update procedure]. The **Source table** and **Source column** fields will provide the Dimension table's side of the join. The options for this field are: False, True, Manual and Pre Join. Setting this field to Manual, changes the way the Dimension table is looked up during the Stage table update procedure build. It enables you to join the dimension manually in the **Source Join** wizard (used to build the 'Where' clause). The usual dialog for matching the dimension business key to a column or columns in the Stage table is not displayed, if this option is enabled. Setting this field to Pre Join, activates the **Pre Join Source Table** field and enables you to select a table from the drop-down list.
Pre Join Source Table	Indicates the table from which the pre joined column was sourced. When the Join option is set to False, this field becomes inactive. When the Join option is set to True or Manual, this field is set to the current table name. When the Join option is set to Pre Join, then you can select the required table from the drop-down list.

© Copyright, WhereScape Software Ltd All rights reserved.

Stage Table Column Transformations

Each Stage table column can have a transformation associated with it. The transformation will be included in the generated procedure and will be executed as part of the procedure update.
The transformation must therefore be a valid SQL construct that can be included in a **Select** statement.

For example we could have a transformation of 'load_order_line.quantity * 0.125' to calculate a tax column of 12.5%. Click the **Transformation** tab to enter a transformation.



Note

Transformations only take effect when the procedure is re-generated. Refer to **Transformations** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

Set Merge Procedure

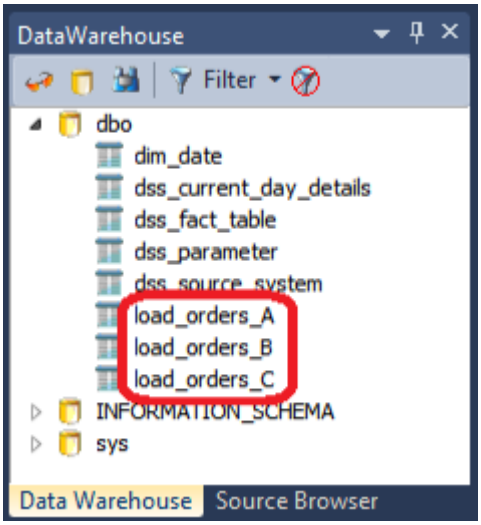
The set merge procedure type allows the merging of two or more identical tables. The tables to be merged must have exactly the same number of columns and column names.

If necessary, additional blank columns could be added to one or other of the tables to make them identical.

To use this procedure you must simply have the tables to be merged mentioned at least once in the **Source Table** field of a column's properties.

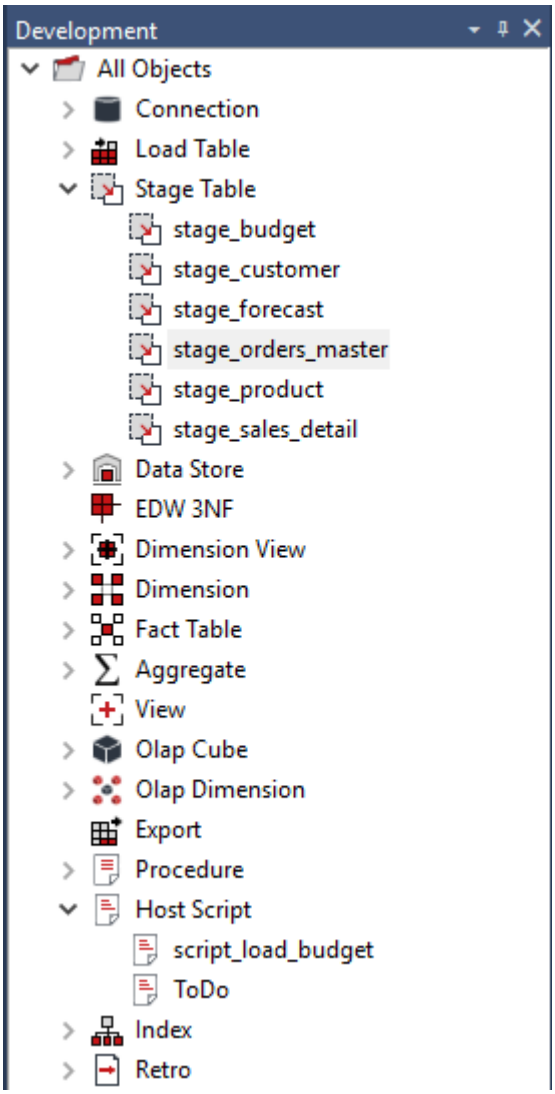
In this example, we will merge three load tables into a single Stage table.

1. The browser pane shows the three Load tables:



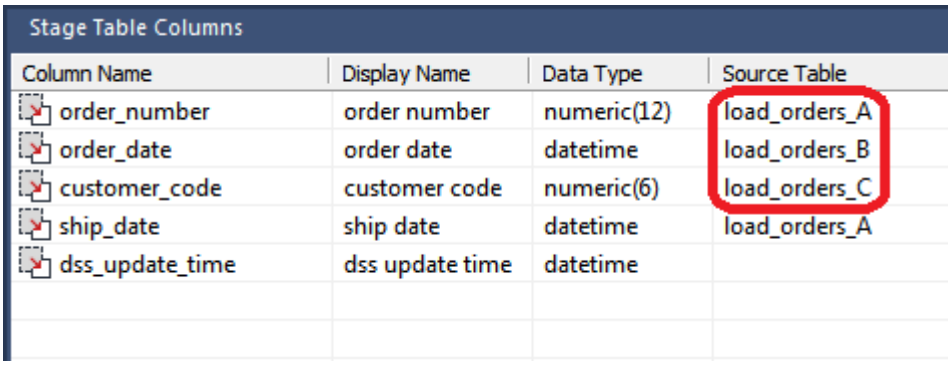
2. Double-click the Stage table object group and then drag one of these Load tables from the source pane on the right, into the Stage Object work area in the middle pane.
    - Name the stage table, for example **stage_orders_master**.

© Copyright, WhereScape Software Ltd All rights reserved.



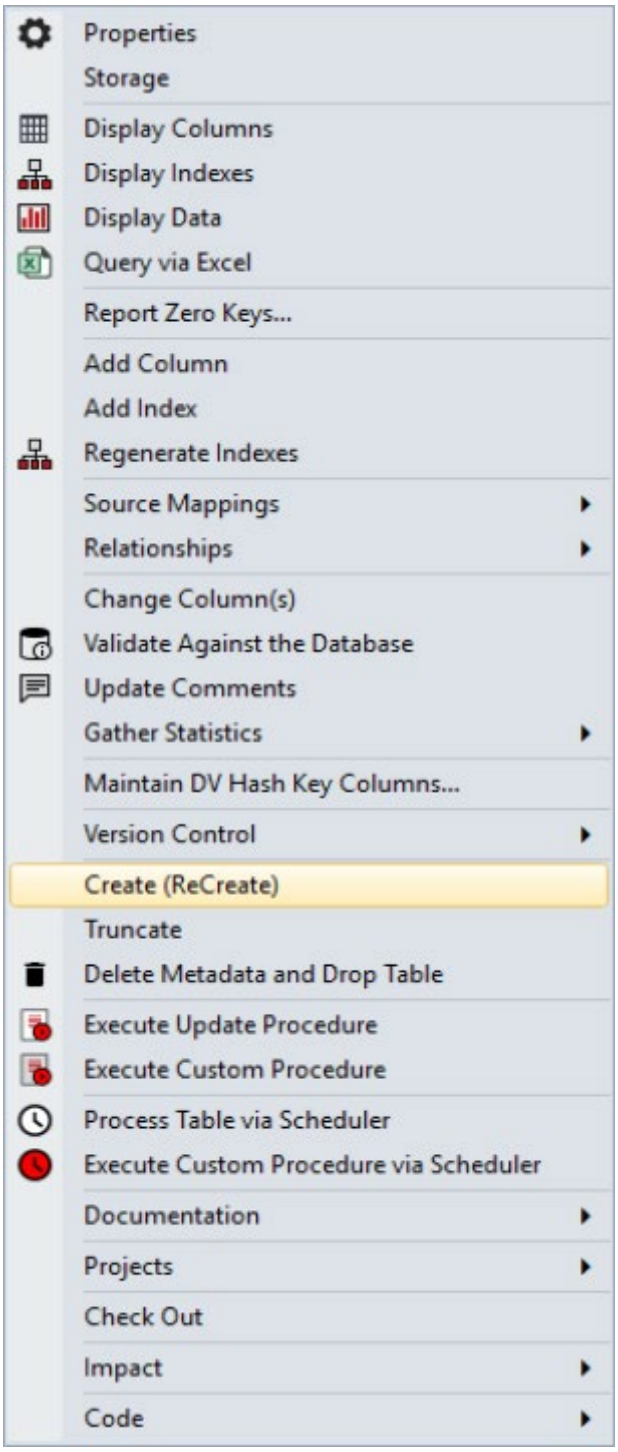
3. Next, modify the source table column to include one instance of each of the three Load tables; the order does not matter.

   You can do this either by typing in the table directly, or by going to each of the column's Properties.



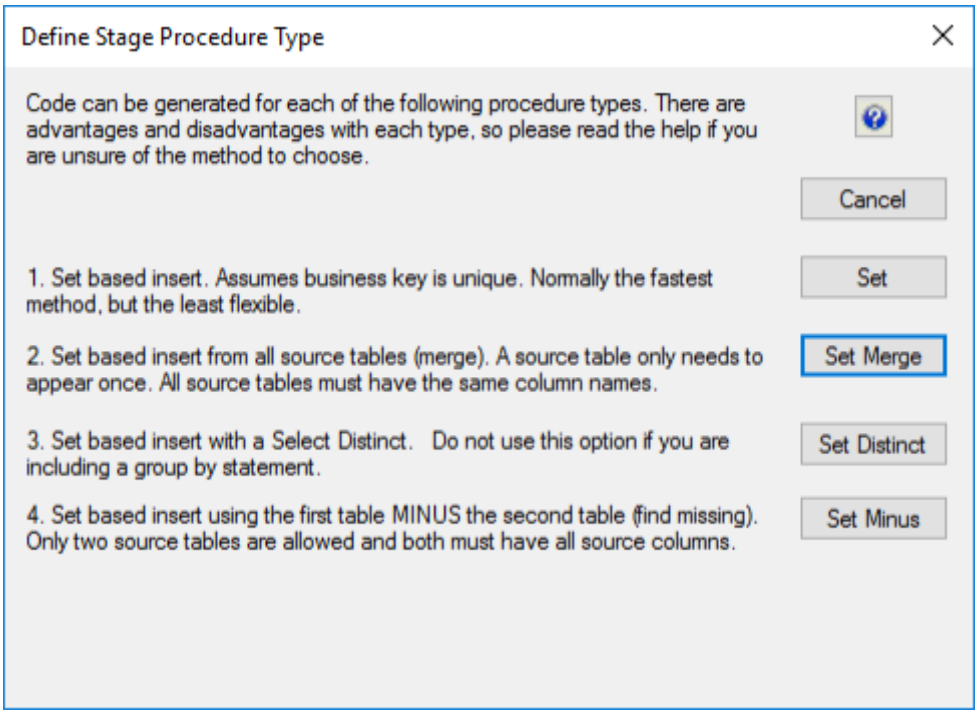
© Copyright, WhereScape Software Ltd All rights reserved.

4. Right-click the Stage table and select **Create (ReCreate)**.



5. Double-click the Stage table in the left pane to bring up the **Properties** window.
   - Click the **Rebuild** button to rebuild the stored procedure.
   - Select **Set Merge** as the procedure type.

© Copyright, WhereScape Software Ltd All rights reserved.



© Copyright, WhereScape Software Ltd All rights reserved.

6. The stored procedure is rebuilt, as follows:

```
-- ================================================================
-- DBMS Name       :    SQL Server
-- Script Name     :    update_stage_orders_Master
-- Description     :    Build the staging table stage_orders_Master
-- Generated by    :    Version 6.5.5.2 (build 111221)
-- Generated for   :    Testing, Tutorials and Documentation
-- Generated on    :    Tuesday, February 07, 2012 at 20:43:10
-- Author          :    QuickStart
-- ================================================================
-- Notes / History
--
CREATE PROCEDURE update_stage_orders_Master
  @p_sequence          integer
, @p_job_name          varchar(256)
, @p_task_name         varchar(256)
, @p_job_id            integer
, @p_task_id           integer
, @p_return_msg        varchar(256) OUTPUT
, @p_status            integer      OUTPUT
AS
  SET XACT_ABORT OFF  -- Turn off auto abort on errors
  SET NOCOUNT ON      -- Turn off row count messages

  --=======================================================
  -- Control variables used in most programs
  --=======================================================
  DECLARE
    @v_msgtext         varchar(255)  -- Text for audit_trail
  , @v_sql             nvarchar(255) -- Text for SQL statements
  , @v_set             integer       -- commit set
  , @v_analyze_flag    integer       -- analyze flag
  , @v_step            integer       -- return code
  , @v_update_count    integer       -- no of records updated
  , @v_insert_count    integer       -- no of records inserted
  , @v_count           integer       -- General counter
  , @v_db_code         varchar(10)   -- Database error code
  , @v_db_msg          varchar(100)  -- Database error message

  --=======================================================
  -- General Variables
  --=======================================================
  DECLARE
    @v_return_status   integer       -- Update result status
  , @v_getkey_status   integer       -- GetKey procedure status
  , @v_row_count       integer       -- General row count
  , @v_truncate_date   datetime      -- Used to set date to midnight
  , @v_dss_source_system_key integer -- Used for dimension joins
  , @v_dss_current_flag      char(1) -- Used for dimension joins
  , @v_dss_update_time       datetime  -- Used for date insert



  --=======================================================
  -- MAIN
  --=======================================================
  SELECT @v_step = 0
  BEGIN TRY

  -- Set initial variable values
  SELECT
    @v_set           = 0
  , @v_analyze_flag  = 0
  , @v_step          = 1
  , @v_update_count  = 0
  , @v_insert_count  = 0
  , @v_dss_source_system_key = 1
  , @v_dss_current_flag = 'Y'
  , @v_dss_update_time = GETDATE()
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
--------------------------------------------------------
-- Delete all existing data from the staging table
--------------------------------------------------------
SET @v_sql = N'TRUNCATE TABLE stage_orders_Master'
EXEC @v_return_status = sp_executesql @v_sql
IF @v_return_status <> 0
BEGIN
  SET @v_db_code = CONVERT(varchar, @v_return_status)
  SELECT @v_db_msg = description FROM master.dbo.sysmessages
  WHERE error = @v_return_status
  EXEC WsWrkAudit
      @p_status_code = 'W'
    , @p_job_name    = @p_job_name
    , @p_task_name   = @p_task_name
    , @p_sequence    = @p_sequence
    , @p_message     = 'Failure when truncating table'
    , @p_db_code     = @v_db_code
    , @p_db_msg      = @v_db_msg
    , @p_task_key    = @p_task_id
    , @p_job_key     = @p_job_id

  SET @p_return_msg = ' Failure to truncate stage table. '
    + @v_db_msg
  SET @p_status = -2
  RETURN 0
END

--------------------------------------------------------
-- Insert input records into stage_orders_Master
--------------------------------------------------------
SET @v_step = 100

BEGIN TRANSACTION

INSERT INTO stage_orders_Master WITH ( TABLOCK )
(
  order_number,
  order_date,
  customer_code,
  ship_date,
  dss_update_time
)
SELECT
  load_orders_A.order_number,
  load_orders_A.order_date,
  load_orders_A.customer_code,
  load_orders_A.ship_date,
  @v_dss_update_time
FROM
  load_orders_A

SELECT
  @v_return_status = @@ERROR
, @v_insert_count = @v_insert_count + @@ROWCOUNT

IF @v_return_status <> 0
BEGIN
  ROLLBACK
  SET @p_return_msg = 'Failed Insert into table stage_orders_Master'
  SET @v_db_code = CONVERT(varchar, @v_return_status)
  SELECT @v_db_msg = description FROM master.dbo.sysmessages
  WHERE error = @v_return_status
  EXEC WsWrkAudit 'E', @p_job_name, @p_task_name, @p_sequence,
    @p_return_msg, @v_db_code, @v_db_msg, @p_task_id, @p_job_id
  SET @p_status = -2
  RETURN 0
END
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
COMMIT

------------------------------------------------------------
-- Insert input records into stage_orders_Master
------------------------------------------------------------
SET @v_step = 100

BEGIN TRANSACTION

INSERT INTO stage_orders_Master WITH ( TABLOCK )
(
  order_number,
  order_date,
  customer_code,
  ship_date,
  dss_update_time
)
SELECT
  load_orders_B.order_number,
  load_orders_B.order_date,
  load_orders_B.customer_code,
  load_orders_B.ship_date,
  @v_dss_update_time
FROM
  load_orders_B

SELECT
  @v_return_status = @@ERROR
, @v_insert_count = @v_insert_count + @@ROWCOUNT

IF @v_return_status <> 0
BEGIN
  ROLLBACK
  SET @p_return_msg = 'Failed Insert into table stage_orders_Master'
  SET @v_db_code = CONVERT(varchar, @v_return_status)
  SELECT @v_db_msg = description FROM master.dbo.sysmessages
  WHERE error = @v_return_status
  EXEC WsWrkAudit 'E', @p_job_name, @p_task_name, @p_sequence,
    @p_return_msg, @v_db_code, @v_db_msg, @p_task_id, @p_job_id
  SET @p_status = -2
  RETURN 0
END

COMMIT
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
--------------------------------------------------------
-- Insert input records into stage_orders_Master
--------------------------------------------------------
SET @v_step = 100

BEGIN TRANSACTION

INSERT INTO stage_orders_Master WITH ( TABLOCK )
(
  order_number,
  order_date,
  customer_code,
  ship_date,
  dss_update_time
)
SELECT
  load_orders_C.order_number,
  load_orders_C.order_date,
  load_orders_C.customer_code,
  load_orders_C.ship_date,
  @v_dss_update_time
FROM
  load_orders_C

SELECT
  @v_return_status = @@ERROR
, @v_insert_count = @v_insert_count + @@ROWCOUNT

IF @v_return_status <> 0
BEGIN
  ROLLBACK
  SET @p_return_msg = 'Failed Insert into table stage_orders_Master'
  SET @v_db_code = CONVERT(varchar, @v_return_status)
  SELECT @v_db_msg = description FROM master.dbo.sysmessages
  WHERE error = @v_return_status
  EXEC WsWrkAudit 'E', @p_job_name, @p_task_name, @p_sequence,
    @p_return_msg, @v_db_code, @v_db_msg, @p_task_id, @p_job_id
  SET @p_status = -2
  RETURN 0
END

COMMIT


--------------------------------------------------------
-- All Done report the results
--------------------------------------------------------
```

© Copyright, WhereScape Software Ltd All rights reserved.

Data Store Objects

Data Store Object Overview

A Data Store Object is a data warehouse object used to store any type of data for later processing.
In WhereScape RED, Data Store objects have many of the code generating attributes of stage, dimension and fact tables. Data Store objects can be thought of as a source system for the data warehouse.
Alternatively, they may be reported off directly by users and reporting tools. Data Store Objects can be considered either reference or transactional in nature.

A Data Store Object is built from the Data Warehouse connection. Unless you are retrofitting an existing system, Data Store Objects are typically built from one or more load or stage tables.
The Data Store model may be retrofitted from an enterprise modeling tool. Refer to **Importing a Data Model** for details.

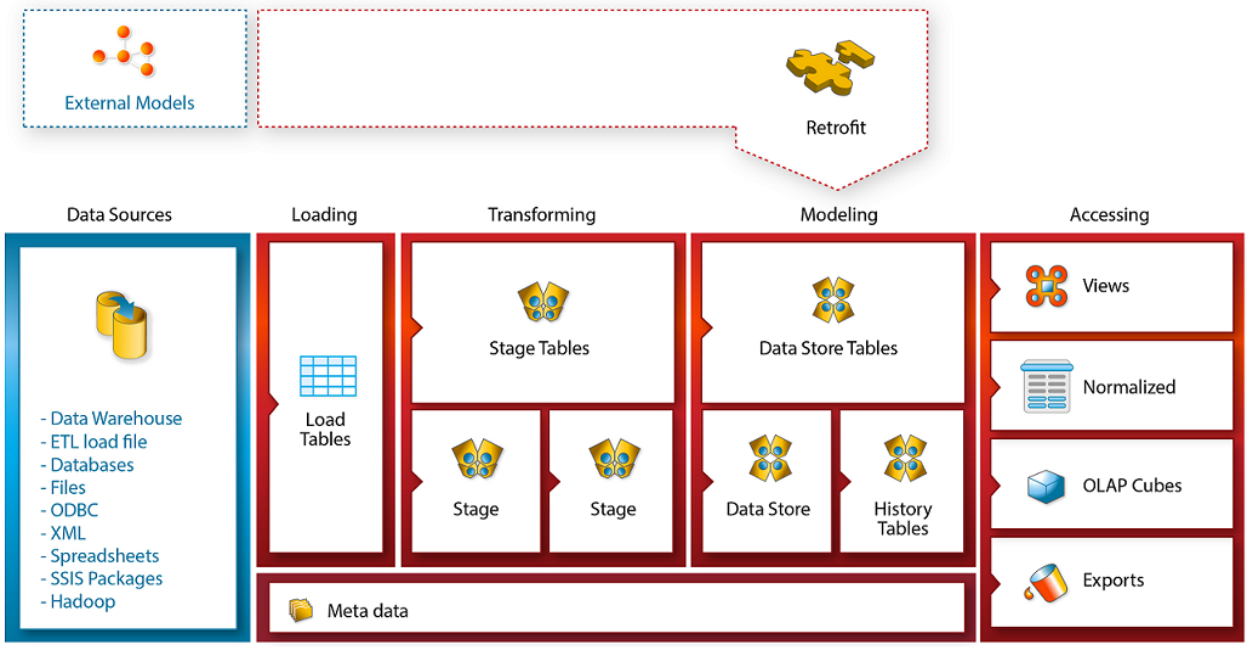
The usual steps for creating a Data Store model are defined below and are covered in this chapter.

The steps are:

1. Identify the source reference or transactional data that will constitute the Data Store Object. If the data is sourced from multiple tables ascertain if a join between the source tables is possible, or if one or more intermediate stage (work) tables would be a better option.
2. Using the 'drag and drop' functionality drag the load or stage table that is the primary source of information for the Data Store Object into a Data Store target. Refer to **Building a Data Store Object** for details.
3. If there's only one source table and all of the columns from it are being used, you can select the auto create option to build and load the table. This automatically completes the next four steps. Refer to **Building a Data Store Object** for details..
4. Add columns from other Load and/or Stage tables if required. Refer to **Building a Data Store Object** for details.
5. Create the Data Store Object in the database. Refer to **Building a Data Store Object** for details.
6. Build the update procedure. Refer to **Generating the Data Store Update Procedure**.
7. Run the update procedure and analyze the results.

If necessary, modify the update procedure or create a custom procedure.

WhereScape RED Overview: Data Store



© Copyright, WhereScape Software Ltd All rights reserved.

Data Store Object Keys

Data Store Objects have Business Keys, they do not usually have Artificial Keys.

Business Key

The business key is the column or columns that uniquely identify a record within a Data Store Object.

Where the Data Store Object maps back to a single or a main table in the source system, it is usually possible to ascertain the business key by looking at the unique keys for that source table.

The business key is sometimes referred to as the 'natural' key. Examples of business keys are:

- The product SKU in a product table
- The customer code in a customer table
- The IATA airport code in an airport table.

It is assumed that business keys will never be NULL. If a null value is possible in a business key then the generated code will need to be modified to handle the null value by assigning some default value.
In the following examples, the business key column is modified by using a database function and default value:

- **DB2 and Teradata:** COALESCE(business_key,'N/A')
- **SQL Server:** ISNULL(business_key,'N/A')
- **Oracle:** NVL(business_key,'N/A')

Note

Business keys are assumed to never be Null. If they can be null it is best to transform them to some value prior to the Data Store or stage table update.
If this is not done, an un-modified update procedure will probably fail with a duplicate key error on the business key index.

© Copyright, WhereScape Software Ltd All rights reserved.

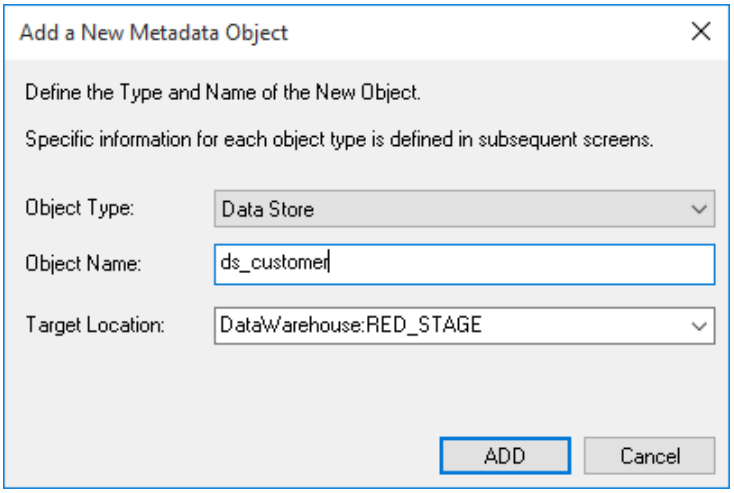
Building a Data Store Object

Data Store Objects are often sourced from one table in the base application.

The process for building a Data Store Object begins with the drag and drop of the load or stage table that contains the bulk of the Data Store Object's information.

Drag and Drop

1. Create a Data Store Object target by double-clicking on the **Data Store** group in the left pane. The middle pane will display a list of all existing Data Store Objects in the current project. When such a list is displayed in the middle pane, the pane is identified as a Drop Target for new Data Store Objects.
2. Browse to the Data Warehouse via the **Browse > Data Warehouse** menu option.
3. Drag the Load or Stage table that contains the bulk of the Data Store Object columns into the middle pane. Drop the table anywhere in the pane.
4. The new object window appears and identifies the new object as a Data Store Object and provides a default name, based on the name of the selected Load or Stage table.
5. Either accept this name or type in the name of the Data Store Object and click **ADD** to proceed:



Data Store Object Properties

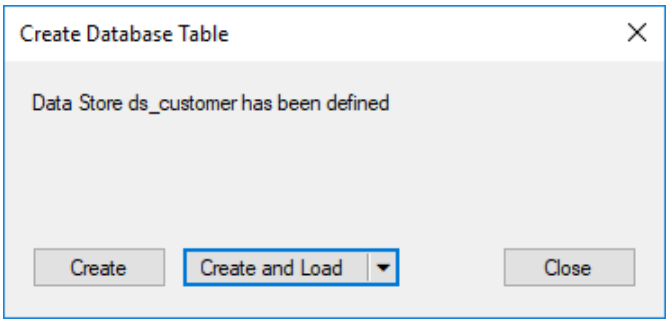
The table Properties window for the new table is displayed.

- If required, the Data Store Object can be changed to be a history table by choosing History from the table type drop-down list on the right side of the dialog. History tables are like slowly changing dimensions in dimensional data warehouses. Refer to **Building a Dimension** for details. Change the storage options if desired.
- If prototyping, and the Data Store Object is simple (e.g. one source table) then it is possible to create, load and update the Data Store Object in a couple of steps.
- If you want to do this, select the **(Build Procedure...)** option from the **Update Procedure** drop-down, and answer **Create and Load** to the next question.

© Copyright, WhereScape Software Ltd All rights reserved.

Create and Load

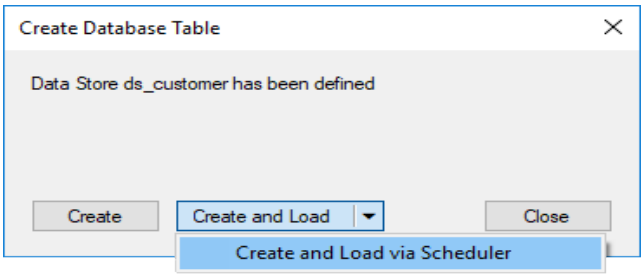
If you chose to build the update procedure the following window appears, after clicking **OK** on the Properties window. It asks if you want to create the Data Store table in the database and execute the update procedure.



If **Create** or **Create and Load** is selected and a new procedure creation was chosen, proceed directly to the **Generating the Data Store Update Procedure**.
If you have additional columns to add or columns to delete then select **Close** and proceed as follows below.

Note

It is possible to create and load the table via the Scheduler; by selecting this option from the drop-down list on the **Create and Load** button:



Deleting and Changing columns

The columns defined for the Data Store Object is displayed in the middle pane.

- It is possible to delete any unwanted columns by highlighting a column name or a group of names and clicking **Delete**.
- The name of a column can also be changed by by right-clicking the column and choosing **Properties** to edit its properties.
- Any new name must conform to the database naming standards. Good practice is to use alphanumerics and the underscore character.
  Refer to **Data Store Column Properties** for a more details.

Tip:

When prototyping and in the initial stages of an analysis area build, it is best not to remove columns, nor to change their names to any great extent. This type of activity is best left, and done after end users have used the data and provided feedback.

Adding columns

With the columns of the Data Store Object displayed in the middle pane, this pane is considered a drop target for additional columns.

- It is a simple matter to select columns from other load and/or stage tables and drag these columns into the middle pane.
- The source table column in the middle pane shows where each column was dragged from.

© Copyright, WhereScape Software Ltd All rights reserved.

- The column **description** could be acquired from three different tables. Best practice is to rename at least two of the columns, perhaps also adding context to the column name. For example, description could become group_description, and so forth.
- There are a number of WhereScape RED ancillary columns that do not have a source table. These columns have been added by WhereScape RED, and are added depending on earlier choices.

A description of these columns follows:

Column name	Description
dss_start_date	Used for history tables. This column provides a date time stamp when the Data Store Object record came into existence. It is used to ascertain which Data Store Object record should be used when multiple are available.
dss_end_date	Used for history tables. This column provides a date time stamp when the Data Store Object record ceased to be the current record. It is used to ascertain which Data Store Object record should be used when multiple are available.
dss_current_flag	Used for Data Store history tables. This flag identifies the current record where multiple versions exist.
dss_source_system_key	Added to support history tables that cannot be fully conformed, and the inclusion of subsequent source systems. See the ancillary settings section for more details.
dss_version	Used for Data Store history tables. This column contains the version number of a Data Store history tables record. Numbered from 1 upwards with the highest number being the latest or current version. It forms part of the unique constraint for the business key of a Data Store history tables.
dss_update_time	Indicates when the record was last updated in the data warehouse.
dss_create_time	Indicates when the record was first created in the data warehouse

Creating the table

Once the Data Store Object has been defined in the metadata, we need to physically create the table in the database.

1. This is done by right-clicking on the Data Store Object in the left pane and selecting **Create (ReCreate)** from the pop-up menu.
2. The Results pane displays the results of the creation. A message confirms that the Data Store Object was created. A copy of the actual database create statement and if defined, the results of any index create statements is listed. For the initial create, no indexes will be defined.
3. If the table was not created, then ascertain and fix the problem. A common problem is a 'Duplicate column' where a column has the same name in two of the source tables. The best way to find such a column is to double-click the list heading **Column Name**, which sorts the column names into alphabetical order.
4. Another double-click on the heading sorts the columns back into their create order.

The next section covers **Generating the Data Store Update Procedure**.

© Copyright, WhereScape Software Ltd All rights reserved.

Generating the Data Store Update Procedure

Once a Data Store object has been defined in the metadata and created in the database, an update procedure can be generated to handle the joining of any tables and the update of the Data Store object.

Notes

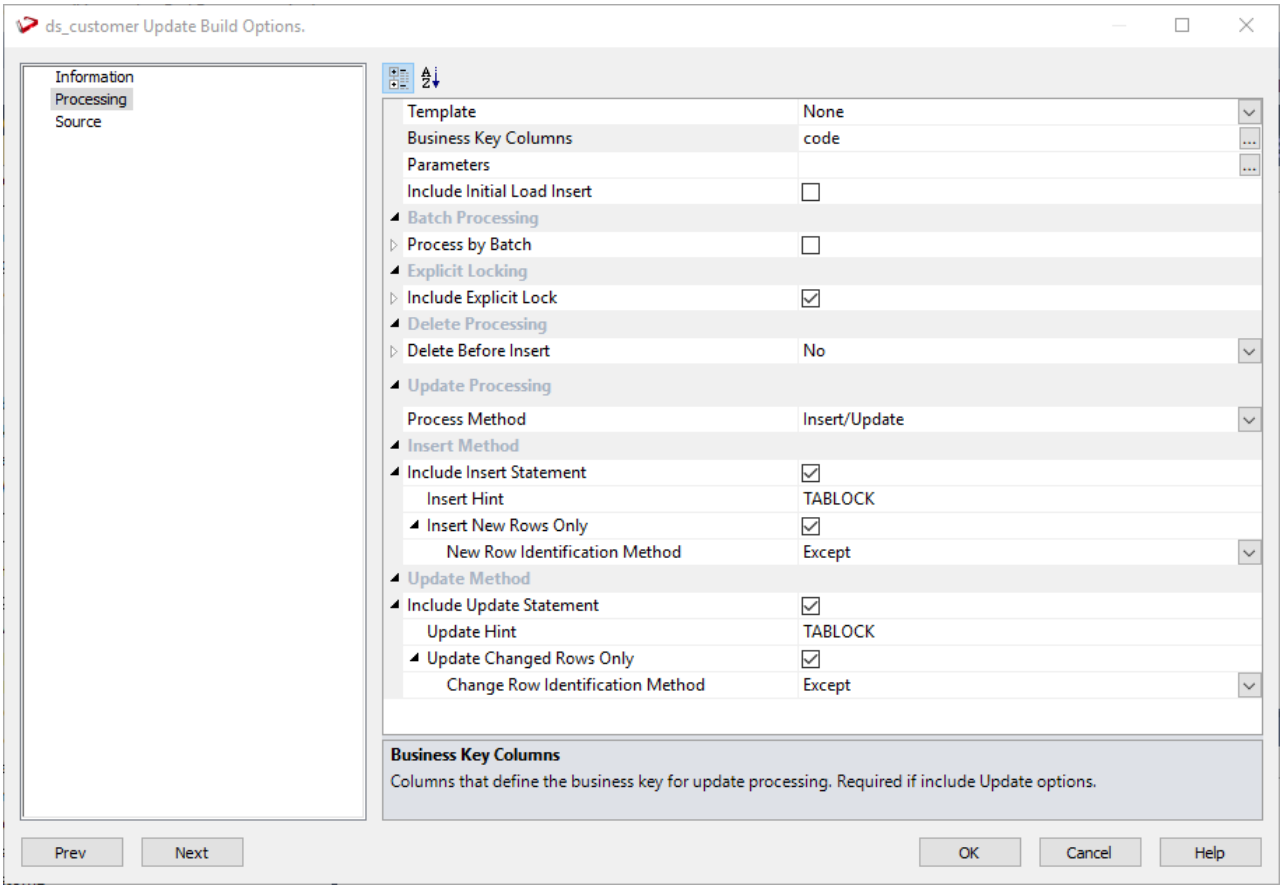
- You can also generate an update procedure via a template, refer to **Rebuilding Update Procedures** for details.
- If the Data Store object is created in a custom database target, then an additional processing option is available—PowerShell script-based processing. Refer to **Script Templates for Custom Database Table Objects** for details.

Generating a Procedure

To generate a procedure:

1. Right-click the Data Store object in the left pane and select **Properties**.
2. Click the **Rebuild** button to start the process of generating the new procedure.
3. A series of options are presented.

Processing Tab

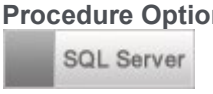
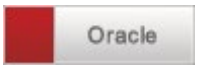
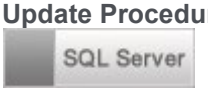
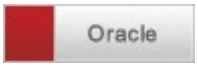
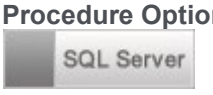
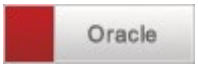


Fields	Description
Template	Enables you to generate update procedures via a **template**.
Business Key Columns	Columns that define the business key for update processing. This is required for include Update options. • Clicking on the ellipsis button will bring up the **Business Key** selection screen.

© Copyright, WhereScape Software Ltd All rights reserved.

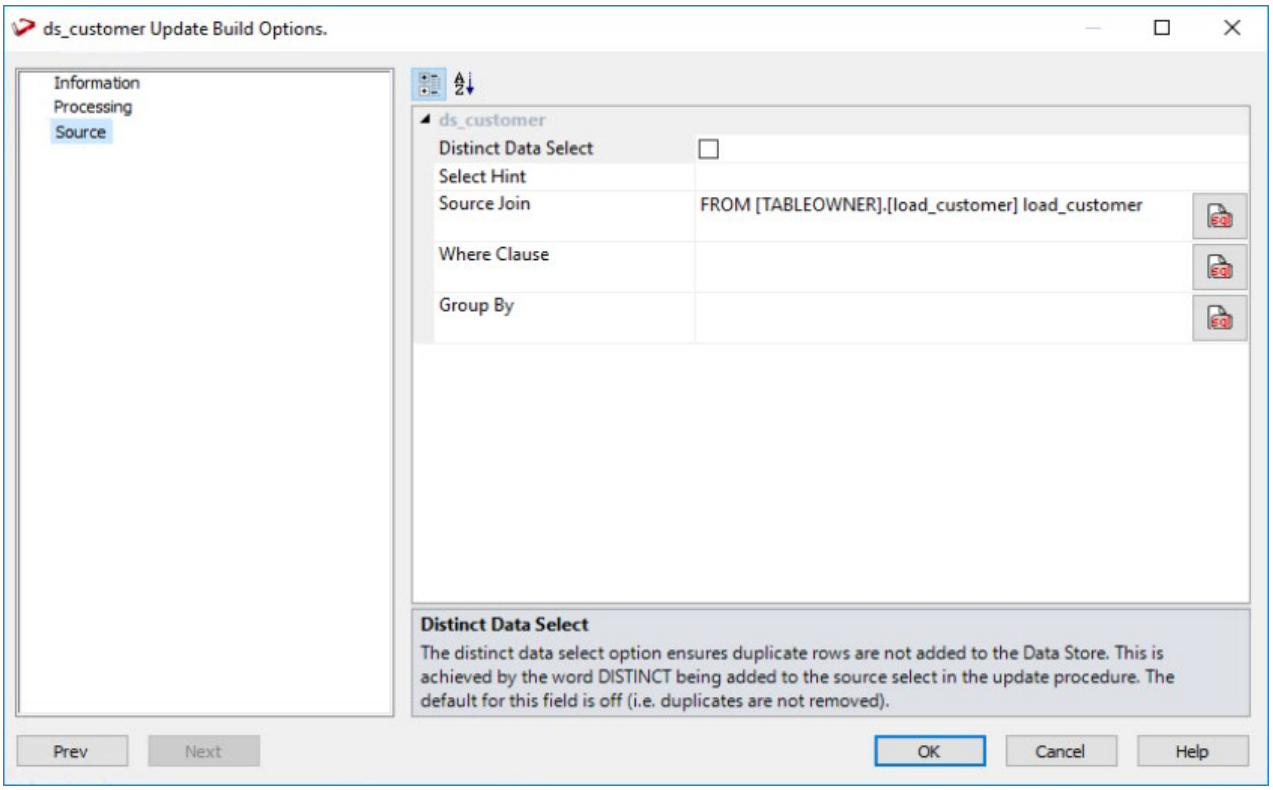
Fields	Description
	• A business key can be made up of multiple columns, but it must provide a unique identifier. Where multiple columns separately uniquely identify rows in the Data Store object, choose one to act as the primary business key. For example, a source table may have a unique constraint on both a product code and a product description. Therefore, the description as well as the code must be unique. It is of course possible to combine the two columns, but the normal practice would be to choose the code as the business key. **Tip** • Use the column name ascending/descending buttons to sort column names. To revert to the meta column order, click on the meta column order button. • **NULL Values** - none of the columns chosen as the business key should ever contain a NULL value. See the note at the start of the Dimensions chapter.
Parameters	Any parameters selected are included in the generated update procedure as variables. The procedure will include code to retrieve the value of the parameter at run time and store it in the declared variable. Clicking on the ellipsis button opens the **Parameters** selection screen. The variables can also be used in column transformations and in the from/where clause for the update procedure. Some databases have a 30 character limit for variable names. WhereScape RED ensures the variables added for any parameters are less than 30 characters long by creating variable names in the form v_ followed by the first 28 characters of the parameter name. For example, a parameter called MINIMUM_ORDER_NUMBER_SINCE_LAST_SOURCE_LOAD will be available as the variable v_MINIMUM_ORDER_NUMBER_SINCE_L. **Tip** • WhereScape RED parameters should be unique within the first 28 characters to avoid conflicting variables names. • If the desired parameter doesn't exist in the metadata yet, a new parameter can be added by clicking on the **Add New** button on the bottom leftmost corner of the Select Parameters dialog. • Refer to **Parameters** for more information.
Enable Parallel DML	**Oracle Only**. Adds all the code required to the update procedure to enable Oracle parallel inserts. Default for this option is not set.
Include Initial Load Insert	Adds an additional insert statement to the update procedure that runs if the target Data Store is empty. The benefit of this is improved performance inserting into an empty table without performing any checks to see if rows already exist. The default for this field is off (i.e. an initial insert statement is not added to the procedure).
Process by Batch	Allows users to select a column to drive data processing in a loop based on the distinct ordered values of the selected columns. The update procedure loops on this column and performs the delete, update and/or insert for each value. If the column chosen is a date datatype (date, datetime or timestamp), then the user is able to specify yearly, monthly, daily or column level looping. The default for this field is off (do not do batch processing).
Delete Before Insert	Allows selection of how to process deletes. It enables a delete statement to be added to the update procedure before any update or insert statement. This is a particularly useful option for purging old data and for updates based on a source system batch number. If this option is selected, the following options are also available: • **Truncate Option** – Optional **Oracle** TRUNCATE clause which is appended to the truncate statement. Add a truncate option such as REUSE/DROP STORAGE by typing it in the truncate options box.

© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
Issue Warning if a Delete occurs	This option sets the procedure to a warning state if any deletes occur.
Delete Where Clause	The delete where clause is appended to the generated delete statement to constrain the rows deleted.
Process Method	Select between Insert/Update and Merge which enables you to use the Merge statement instead of two separate Insert and update statements.
Source Table Locking	Teradata only. Enables a locking request modifier to be specified for each source table. The specified locking request modifier is applied to each source table during generated update procedures. By default this is set to 'ACCESS' which locks each row being accessed, a blank entry will result in no locking clause in the generated procedure.
Include Insert Statement	Includes the insert statement in the procedure to insert new rows in the Data Store.
Insert Hint	Enter a database hint to be used in the INSERT statement. This is an Oracle and SQL Server only option. Defaults can be configured in **Tools > Options > Default Update Procedure Options**.  Default is TABLOCK.  Default is APPEND.
Insert New Rows Only	Uses change detection to work out what rows require inserting.
New Row Identification Method	Method used to identify that records in source are not currently recorded in the target table.
Existing Data Selection Hint	Database-compliant hint to be used for the existing data select statement.
Include Update Statement	Includes an update statement in the procedure to update changing rows in the Data Store. If this option is chosen, then the Update Changed rows only option is available.
Include Hint	Enter a database hint to be used in the INCLUDE statement. This is an Oracle and SQL Server only option. Defaults can be configured in **Tools > Options > Default Update Procedure Options**.  Default is TABLOCK.  Default is APPEND.
Update Changed Rows Only	Uses change detection to work out what rows require updating. Choosing this option, enables the **Change Row identification Method**.
Change Row Identification Method	Method used to identify that records in source have changed from what is currently recorded in the target table.
Existing Data Selection Hint	Database-compliant hint to be used for the existing data select statement.
Merge Hint	Enter a database hint to be used in the MERGE statement. This is an Oracle and SQL Server only option. Defaults can be configured in **Tools > Options > Default Update Procedure Options**.  Default is TABLOCK.  Default is APPEND.

© Copyright, WhereScape Software Ltd All rights reserved.

Source Tab



Fields	Description
Distinct Data Select	Ensures duplicate rows are not added to the Data Store. This is achieved by the word DISTINCT being added to the source select in the update procedure. The default for this field is not set.
Source Join	The From clause, including Source Join information.
Where Clause	The Where clause.
Group By	The Group By clause.

Simple Join

A simple join only returns rows where data is matched in both tables. So for example if table A has 100 rows and table B has a subset of 24 rows. If all the rows in table B can be joined to table A then 24 rows will be returned. The other 76 rows from table A will not be returned.

Outer Join

The outer join returns all rows in the master table regardless of whether or not they are found in the second table. So if the example above was executed with table A as the master table then 100 rows would be returned. 76 of those rows would have null values for the table B columns.

Notes

- When WhereScape RED builds up an outer join, it needs to know which table is the master table, and which is subordinate. Select the join column from the master table first. In the example screen above the table 'load_order_header' has had its column chosen and the column for the table 'load_order_line' is currently being chosen. This will result in the 'load_order_header' table being defined as the master, as per the example statement above. The results of this example select are that a row will be added containing order information regardless of whether or not a corresponding load_order_line entry exists.
- When upgrading from a RED version previous to 6.8.2.0 and moving existing objects to a target location, all procedures that reference those objects will need to be rebuilt.
  Any **FROM** clauses will also need to be manually regenerated in order for the table references to be updated to the new [TABLEOWNER] form.

© Copyright, WhereScape Software Ltd All rights reserved.

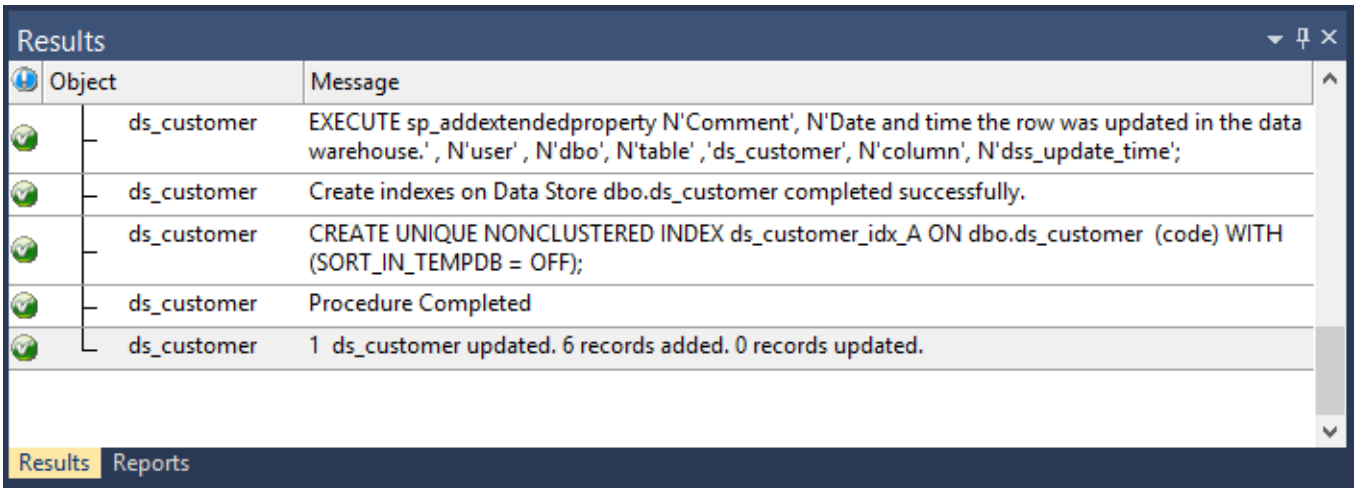
Building and Compiling the Procedure

- Once the relevant options are completed, click **OK**. The procedure is built and compiled.
- If the compile fails, an error is displayed along with the first few lines of error messages.
- Compile fails typically occur when the physical creation of the table was not done. If the compile fails for some other reason, the best approach is to use the procedure editor to edit and compile the procedure.
- The procedure editor highlights all the errors within the context of the procedure.
- Once the procedure has been successfully compiled, it can either be executed interactively or passed to the scheduler.

Indexes

By default, a number of indexes is created to support each Data Store object.

These indexes are added, once the procedure has been built. An example of the type of indexes created is as follows:



Additional indexes can be added, or these indexes changed. Refer to **Indexes** for details.

Data Store Artificial Keys

By default, Data Store Objects in WhereScape RED do not have an artificial (surrogate) key.

Artificial keys can be added manually but if needed could indicate Data Store Objects are not the correct WhereScape RED object for this table (perhaps an EDW 3NF Table would be more appropriate).

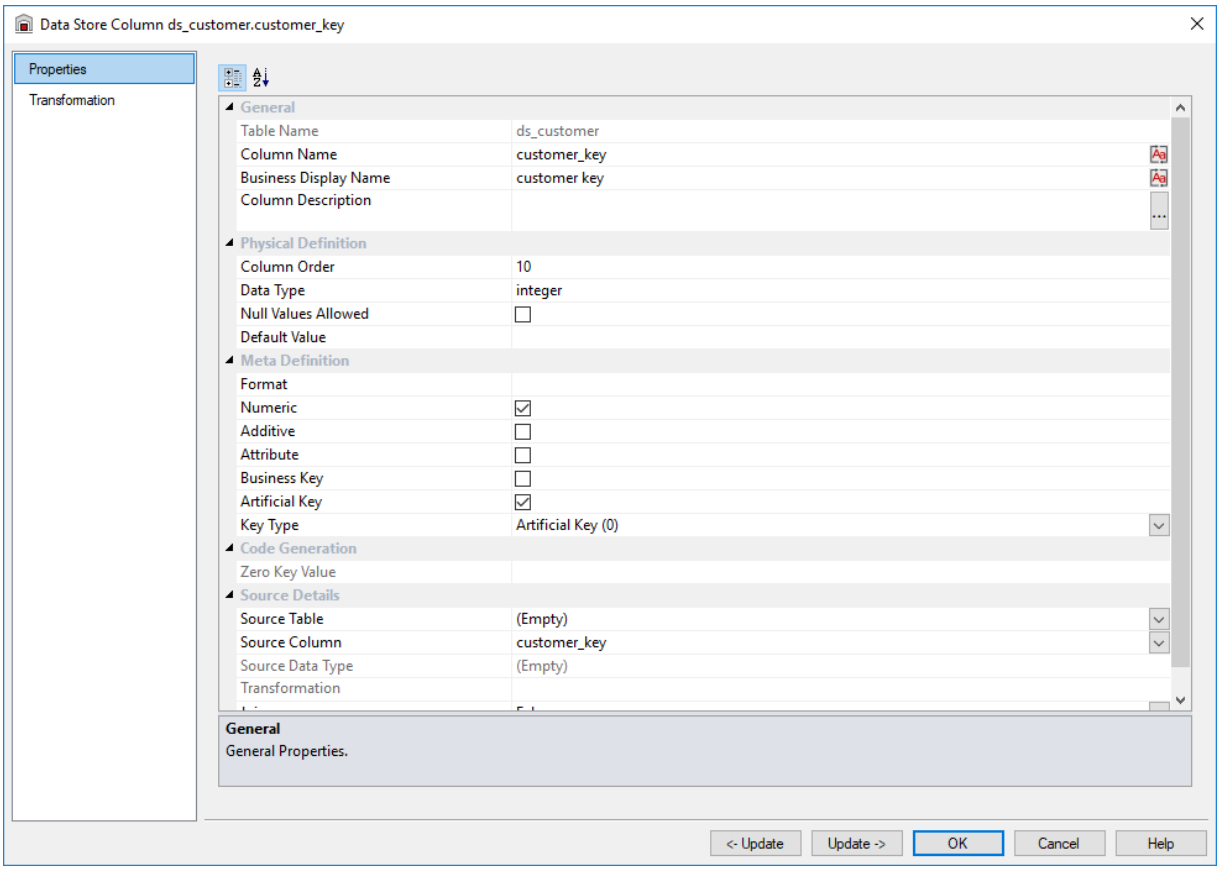
Edit the properties of the new column to have the correct name and order, source table and column, data type, key type and flags.

Specifically:

1. Right click a column in the middle pane and select either **Add Column** or **Duplicate Column** from the context menu.
2. Edit the properties of the new column to have the correct name and order, source table and column, data type, key type and flags as below:
   - The **Column Name** and **Source Column** must be the same.
   - The **Source Table** must be empty.
   - The **Data Type** should be:
     - **DB2:** integer generated by default as identity (start with 1, increment by 1)
     - **Oracle:** integer
     - **SQL Server:** integer identity(0,1)
     - **Teradata:** integer
   - The **Key Type** must be 0.
3. Only the **Numeric** and **Artificial Key** options should be enabled.

   The following example shows a manually added artificial key column:

© Copyright, WhereScape Software Ltd All rights reserved.



The artificial key for a Data Store Object is set via a sequence in Oracle and an identity column in SQL Server and DB2. This artificial key normally, and by default, starts at one and progresses as far as is required.

A WhereScape standard for the creation of special rows in the Data Store tables is as follows:

Key value	Usage
1 upwards	The standard artificial keys are numbered from 1 upwards, with a new number assigned for each distinct Data Store Object record.
0	Used as a join to the Data Store Object when no valid join existed. It is the convention in the WhereScape generated code that any EDW 3NF table business key that either does not exist or does not match is assigned to key 0.
-1 through -9	Used for special cases. The most common being where an EDW 3NF table is not appropriate for the record. A new key is used rather than 0 as we want to distinguish between records that are invalid and not appropriate.
-10 backward	Pseudo records. In many cases we have to deal with different granularities in our data. For example, we may have a table that contains actual sales at a product SKU level and budget information at a product group level. The product table only contains SKU based information. To be able to map the budget records to the same table, we need to create these pseudo keys that relate to product groups. The values -10 and backwards are normally used for such keys.

Note

To have a surrogate key auto added for Data Store tables, refer to **Global Naming of Key Columns**.

© Copyright, WhereScape Software Ltd All rights reserved.

Data Store Column Properties

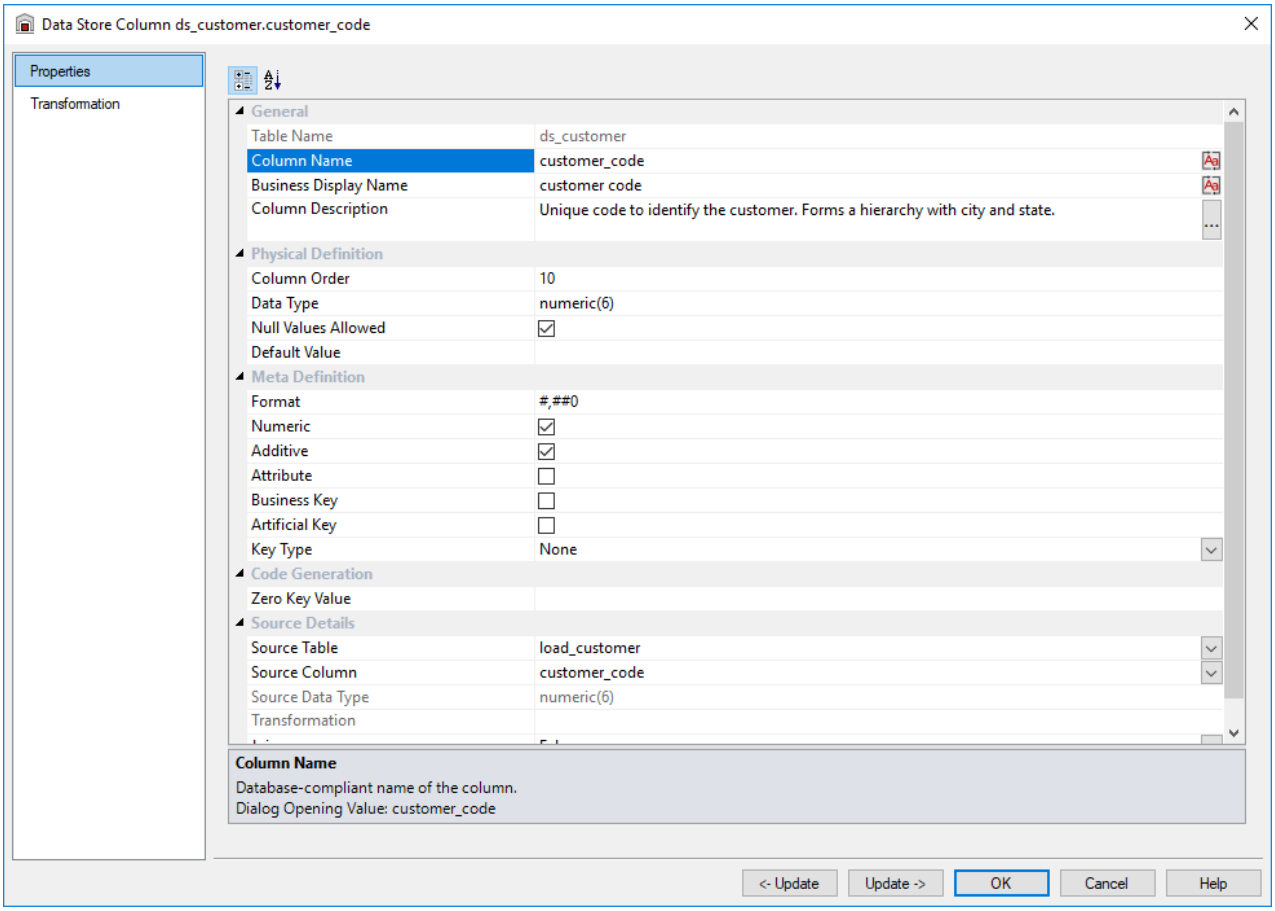
Each Data Store Object column has a set of associated properties. The definition of each property is described below:

- If the **Column name** or **Data type** is changed for a column then the metadata will differ from the table as recorded in the database.
- Use the **Validate/Validate Table Create Status** menu option or the right mouse menu to compare the metadata to the table in the database.
- A right mouse menu option of **Alter table** is available when positioned on the table name after the validate has completed. This option will alter the database table to match the metadata definition.

Tip

- If a database table's definition is changed in the metadata then the table will need to be altered in the database.
- Use the **Validate/Validate Table Create Status** to compare metadata definitions to physical database tables. The option also provides the ability to alter the database table, through a pop-up menu option from the validated table name.

A sample **Properties** screen is as follows:



Tip

- The two special update keys allow you to update the column and step either forward or backward to the next column's properties.
- **ALT-Left Arrow** and **ALT-Right Arrow** can also be used instead of the two special update keys.

© Copyright, WhereScape Software Ltd All rights reserved.

General

Options	Description
Table Name	Database-compliant name of the table that contains the column. [Read-only].
Column Name	Database-compliant name of the column. Typically column-naming standards exclude the use of spaces etc. A good practice is to only use alphanumerics, and the underscore character. Changing this field alters the table's definition. **Note** A case conversion button on the right converts the text between different cases: UPPER CASE, Capitalized Case and lower case. The mode cycles to the next case in the sequence each conversion.
Business Display Name / Column Title	Name that the business uses to refer to the column, which is included in the RED-generated documentation and can be used in the end user layer of other tools. [Does NOT affect the physical database table]. As such it is a free form entry and any characters are valid. **Note** A case conversion button on the right converts the text between different cases: UPPER CASE, Capitalized Case and lower case. The mode cycles to the next case in the sequence each conversion.
Column Description	This field contains the description for the column. It may be a description from a business user's point of view. This field might additionally contain information on where and how the column was acquired. For example if the column is sourced from multiple tables or is a composite or derived column then this definition would normally describe the process used to populate the column. This field is used in the documentation and is available via the view **ws_admin_v_dim_col .** This field is also stored as a comment against the column in the database.

Physical Definition

Options	Description
Column Order	Numeric value that controls the relative order of columns in the database create statement. The lowest numbered column will appear first in the table. Although this affects the physical table definition no action will be taken unless the table is re-created in the database. The columns can be re-numbered based on the existing order by choosing the **Respace Order Number** pop-up menu option when positioned over any column in the table. This action will number the columns in increments of 10 starting at 10. In addition to a simple change of the order field, the column order can be changed by first displaying the columns in the middle pane and then using drag and drop to move the columns around. This drag and drop process will automatically renumber the columns as required.
Data Type	Database-compliant data type that must be valid for the target database. Typical data types for Oracle are integer, number, char(), varchar2() and date. For SQL Server, common types are integer, numeric, varchar() and datetime. See the database documentation for a description of the data types available. Changing this field alters the table's definition.
Null Values Allowed	Determines whether the table column can hold NULL values or whether a value is always mandatory.
Default Value	Initial value that is assigned to the column when a row is inserted into the table but no value is specified for the column.

© Copyright, WhereScape Software Ltd All rights reserved.

Meta Definition

Options	Description
Format	Optional format mask that can be used in end user tools. [Does NOT affect the physical database table]. As such it is a free form entry and any characters are valid. Typically format masks are only used on numeric fields. Example: #,###0.00. It is not worth the effort of populating this field unless it can be utilized by the end user tools in use.
Numeric	Indicates whether the table column holds values that are numeric. This is normally only relevant for fact tables. It does not affect the physical table definition, but rather provides input to the view **ws_admin_v_dim_col** which can be used to assist in the population of an end user tool's end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.
Additive	Indicates whether the table column holds values that are additive. This implies that the column can be summed when performing data grouping in a query. This is normally only relevant for fact tables. It does not affect the physical table definition, but rather provides input to the view **ws_admin_v_dim_col** which can be used to assist in the population of an end user tool's end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.
Attribute	Indicates whether the table column holds values that are descriptive, and/or are used for grouping/summing. An attribute is defined as a column that is non factual. For example we may have an order number, or a invoice number stored in the fact table. Such columns are considered attributes, rather than facts. This checkbox is therefore normally only relevant for fact tables. This checkbox does not affect the physical table definition, but rather provides input to the view **ws_admin_v_dim_col** which can be used to assist in the population of an end user tools end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.
End User Layer Display	Indicates whether the table column is available/visible to end users. If set the documentation will include the column in the glossary and in the user documentation. It is also used to decide what columns appear in the view **ws_admin_v_dim_col**. Typically columns such as the artificial key would not be enabled for end user display.
Business Key	Indicates whether the column is part of the BUSINESS key, which is defined during the update procedure build. [Normally maintained automatically]. Multiple columns can form the primary business key.
Artificial Key	Indicates whether the column is the generated artificial/surrogate key (unique identifier) for the table. Only one artificial key per table is supported. [Normally maintained automatically].
Key Type	Key type that is assigned and used when generating the table's update procedure and indexes. [Normally maintained automatically]. It can be altered here, but this should only be done if the consequences are fully understood and tested. The supported values are: **Key type** / **Meaning** **0** — The artificial key. Set when the key is added during drag and drop table generation. **1** — Component of all business keys. Indicates that this column is used as part of any business key. For example: By default the dss_source_system_key is added to every dimension table. It is considered part of any lookup on that table and has the key type set to 1. Set when the column is added during drag and drop table generation. **2** — Indicates that this column is a dimensional join. Used on fact tables to indicate the dimension keys. Results in an index being built for the column (Bitmap in Oracle). Set during the update procedure generation for a fact table, based on information from the staging table. **3** — Slowly changing column indicator. Used on dimension tables to indicate that the column is being managed as a slowly changing column within the context of a slowly changing dimension. Set when a column is identified during the

© Copyright, WhereScape Software Ltd All rights reserved.

Options	Description	
		dimension update procedure generation.
	4	Slowly changing column indicator. Used on dimension tables to indicate that the column is being managed as a slowly changing column within the context of a slowly changing dimension. Set when a column is identified during the dimension update procedure generation.
	5	Start date of a date ranged dimension. Used on dimension tables to indicate that the column is defined as the starting date for a source system date ranged dimension. Forms part of the business key. Set during the dimension creation.
	6	End date of a date ranged dimension. Used on dimension tables to indicate that the column is defined as the ending date for a source system date ranged dimension. Forms part of the business key. Set during the dimension creation.
	A	Indicates that the column is part of the primary business key. Set whenever a business key is defined as part of an update procedure generation.
	B-Z	Indicates that the column is part of a secondary business key. Only used during index generation and not normally set.

Code Generation

Options	Description
Zero Key Value	Determines the value populated for the column in the **Invalid Join** or **Unknown record**. By default, NULL is used when a value is not specified. All dimensions that use standard WhereScape RED generated procedures have a row with an artificial key of zero. This row is used to link to the fact records when no match on the dimension can be found. For example we may have some sales records that come through without a product defined. WhereScape RED by default associates these fact records with the zero key product dimension entry. Therefore, we might set the zero key value to 'Unknown product' for the name column in the product dimension.

Source Details

Options	Description
Source Table	Identifies the source table where the column's data comes from. This source table is normally a load table, or a dimension table within the data warehouse. If the column was sourced from multiple tables, then the normal practice is to record one of the tables in this field and a comment listing all of the other tables in the **Source strategy** field. This field is used when generating a procedure to update the stage table. It is also used in the track back diagrams and in the documentation.
Source Column	Identifies the source column where the column's data comes from. Such a column is normally a load table column, which in turn may have been a transformation or the combination of multiple columns. This may also be a dimensional key where a dimension is being joined.
Source Data Type	Identifies the source column's data type. [Read-only].
Transformation	Refer to **Dimension Column Transformations** for details. [Read-only].
Join	Indicates whether the table column is used in a table join. [Normally maintained automatically but can be optionally changed to override the default join logic used in the generated update procedure]. The **Source table** and **Source column** fields will provide the dimension table's side of the join. The options for this field are: False, True, Manual and Pre Join. • Setting this field to Manual changes the way the Dimension table is looked up

© Copyright, WhereScape Software Ltd All rights reserved.

Options	Description
	during the Stage table update procedure build. It enables you to join the dimension manually in the **Source Join** wizard (used to build the 'Where' clause). The usual dialog for matching the dimension business key to a column or columns in the Stage table is not displayed, if this option is enabled. • Setting this field to Pre Join activates the **Pre Join Source Table** field and enables you to select a table from the drop-down list.

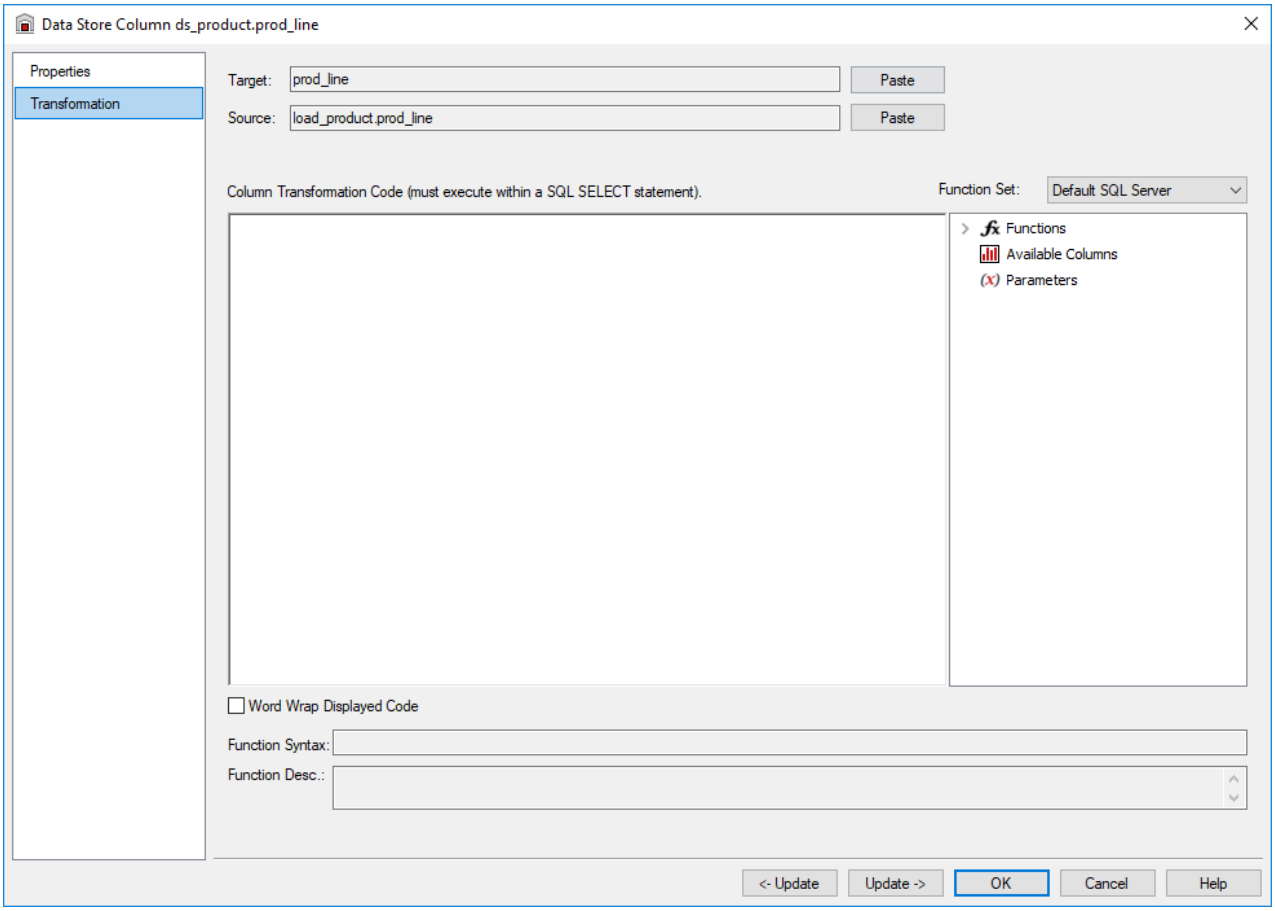
Data Store Object Column Transformations

Each Data Store Object column can have a transformation associated with it. The transformation is included in the generated procedure and is executed as part of the procedure update.
The transformation must therefore be a valid SQL construct that can be included in a **Select** statement.

For example, we could have a transformation of 'load_order_line.qty * 0.125' to calculate a tax column of 12.5%.

Click the **Transformation** tab to enter a transformation.

The transformation screen is as follows:



Note

Transformations are only put into effect when the procedure is re-generated.

Refer to **Transformations** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

EDW 3NF Tables

Note

EDW 3NF/Normalized Table rename

- Former WhereScape RED Normalized Tables have been renamed to EDW 3NF from RED version 6.8.4.3.
  However, this change applies only for new metadata repositories, existing metadata repositories will not be affected and will not have its table's naming modified.
  All references to Normalized tables have been updated in the RED documentation from version 6.8.4.3 onwards. However, some screenshots of the RED left pane browser might still show instances of the Normalized object type instead of the new EDW 3NF type.
- To modify table naming from Normalized to EDW 3NF in existing repositories see **Object Type Names** and **Global Naming Conventions**.
  Please note that short name and table prefixes can be overwritten by the Local Naming conventions setting in the User Preferences. If this is the case, you can disable this option here: **Local Naming conventions**.

EDW 3NF Table Overview

An EDW 3NF Table is a data warehouse object used to build third normal form enterprise data warehouses.

In WhereScape RED, EDW 3NF objects have many of the code generating attributes of Stage, Dimension and Fact tables.

Third normal form enterprise data warehouses can be thought of as a source system for star schema data marts. Alternatively, they may be reported off directly by users and reporting tools.

EDW 3NF tables can be considered either reference or transactional in nature.

An EDW 3NF table is built from the Data Warehouse connection. Unless you are retrofitting an existing system, EDW 3NF Tables are typically built from one or more load or stage tables.

The EDW 3NF model may be retrofitted from an enterprise modeling tool. Refer to **Importing a Data Model** for details.

The usual steps for creating an EDW 3NF model are defined below and are covered in this chapter. The steps are:

1. Identify the source reference or transactional data that will constitute the EDW 3NF Table. If the data is sourced from multiple tables ascertain if a join between the source tables is possible, or if one or more intermediate stage (work) tables would be a better option.
2. Using the 'drag and drop' functionality drag the load or stage table that is the primary source of information for the EDW 3NF Table into an EDW 3NF target. Refer to **Building an EDW 3NF Table** for details.
3. If there's only one source table and all the columns from it are being used, you can select the auto create option to build and load the table. This automatically completes the next four steps. Refer to Building an EDW 3NF Table for details.
4. Add columns from other load and/or stage tables if required. Refer to **Building an EDW 3NF Table** for details.
5. Create the EDW 3NF Table in the database. Refer to **Building an EDW 3NF Table** for details.
6. Build the update procedure. Refer to **Generating the EDW 3NF Update Procedure** for details.
7. Run the update procedure and analyze the results.

If necessary, modify the update procedure or create a custom procedure.

© Copyright, WhereScape Software Ltd All rights reserved.

EDW 3NF Table Keys

EDW 3NF Tables have two types of keys that we will refer to frequently. These are the Business Key and the Artificial Key. A definition of these two key types follows:

Business Key

The business key is the column or columns that uniquely identify a record within an EDW 3NF Table. Where the EDW 3NF Table maps back to a single or a main table in the source system, it is usually possible to ascertain the business key by looking at the unique keys for that source table. The business key is sometimes referred to as the 'natural' key. Examples of business keys are:

- The product SKU in a product table
- The customer code in a customer table
- The IATA airport code in an airport table.

It is assumed that business keys will never be NULL. If a null value is possible in a business key then the generated code will need to be modified to handle the null value by assigning some default value.
In the following examples, the business key column is modified by using a database function and default value:

- **DB2 and Teradata:** COALESCE(business_key,'N/A')
- **SQL Server:** ISNULL(business_key,'N/A')
- **Oracle:** NVL(business_key,'N/A')

Note

Business keys are assumed to never be Null. If they can be null it is best to transform them to some value prior to the EDW 3NF or Stage table update.
If this is not done, an un-modified update procedure will probably fail with a duplicate key error on the business key index.

Artificial Key

By default, EDW 3NF Tables in WhereScape RED do not have an artificial key (artificial keys can be added manually, refer to **EDW 3NF Table Artificial Keys** for details.

An artificial key is the unique identifier that can be used to join an EDW 3NF Table record to other EDW 3NF Tables.
When joining EDW 3NF Tables it would be possible to perform the join using the business key. For EDW 3NF Tables that satisfy one of more of the following conditions, joining with business keys could result in slow query times and excessive use of database storage:

- Multiple column business keys (excessive storage and multiple column joins)
- One or more large character business key columns (excessive storage)
- Very large tables (excessive storage - integer artificial keys often use less space than one small character field)
- History EDW 3NF Tables (complex joins involving a between dates construct)

As query time is one of our key drivers in data warehouse implementations the best answer is often to use some form of artificial key.
A price is paid in the additional processing required doing key lookups, but this is offset by the reduced query times and reduced complexity.

The artificial key is an integer and is built sequentially from 1 upwards.

Refer to the section on artificial keys for a more detailed explanation. An artificial key is sometimes referred to as a "surrogate" key.

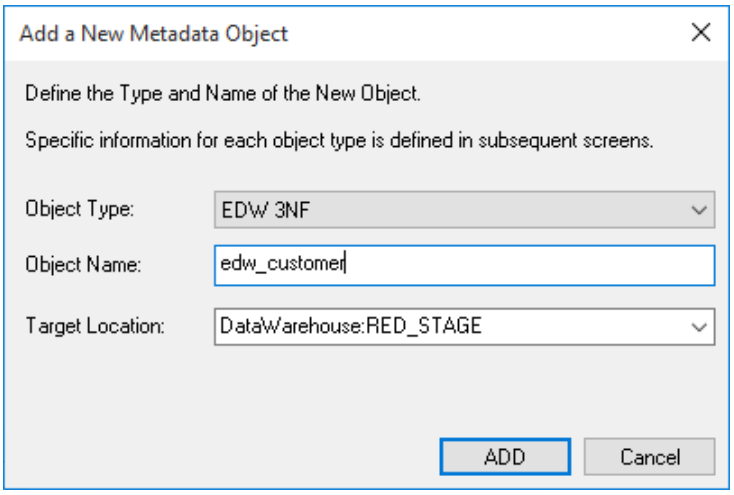
© Copyright, WhereScape Software Ltd All rights reserved.

Building an EDW 3NF Table

EDW 3NF tables are often sourced from one table in the base application. The process for building an EDW 3NF table begins with the drag and drop of the Load or Stage table that contains the bulk of the EDW 3NF table's information.

Drag and Drop

1. Create an EDW 3NF table target by double-clicking the **EDW 3NF** object group in the left pane. The middle pane displays a list of all existing EDW 3NF tables in the current project. When this list is displayed in the middle pane, the pane is identified as a Drop Target for new EDW 3NF tables.
2. Browse to the Data Warehouse via the **Browse > Data Warehouse** menu option.
3. Drag the Load or Stage table, that contains the bulk of the EDW 3NF table columns, into the middle pane. Drop the table anywhere in the pane.
4. The new object window appears and identifies the new object as a EDW 3NF table and provides a default name, based on the name of the selected Load or Stage table.
5. Either accept this name or type in the name of the EDW 3NF table and click **ADD** to proceed:



EDW 3NF Table Properties

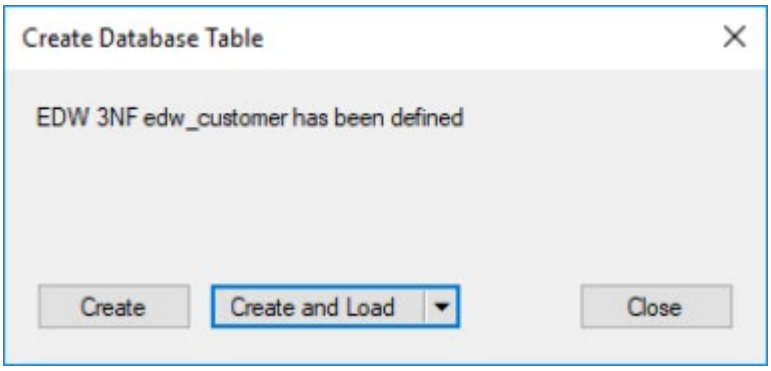
The table Properties window for the new table is displayed.

- If required, the EDW 3NF table can be changed to be a history table by choosing History from the table type drop-down list on the right side of the dialog. History tables are like slowly changing dimensions in dimensional data warehouses. Refer to **Building a Dimension** for details. Change the storage options if desired.
- If prototyping, and the EDW 3NF table is simple (e.g. one source table) then it is possible to create, load and update the EDW 3NF table in a couple of steps.
- If you wish to do this, select the **(Build Procedure...)** option from the **Update Procedure** drop-down, and answer **Create and Load** to the next question.

© Copyright, WhereScape Software Ltd All rights reserved.

Create and Load

If you chose to build the update procedure, the following window appears after clicking **OK** in the **Properties** window. It asks if you want to create the EDW 3NF table in the database and execute the update procedure.

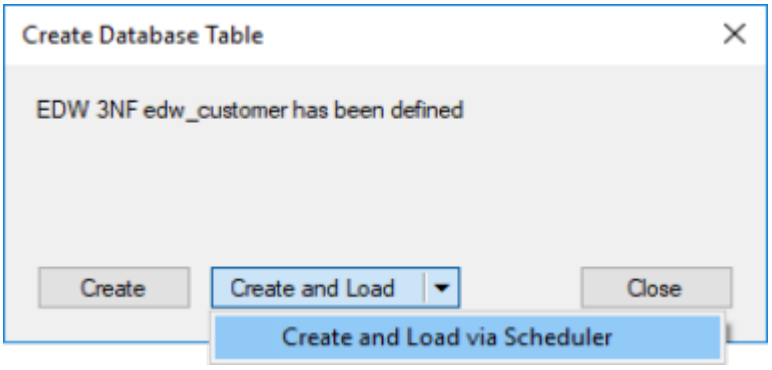


If you are satisfied with the columns that will be used and do not wish to add any additional columns you can select the **Create and Load** button. Alternatively, the **Create** button creates the table in the repository but does not execute an update, enabling you to change columns before loading data into the table.

If **Create** or **Create and Load** is selected and a new procedure creation was chosen, proceed directly to the **Generating the EDW 3NF Update Procedure**.

Note

It is possible to create and load the table via the Scheduler; by selecting this option from the drop-down list on the **Create and Load** button:



If you have additional columns to add or columns to delete then select **Close** and proceed as follows below.

Deleting and Changing columns

The columns defined for the EDW 3NF table is displayed in the middle pane.

- It is possible to delete any unwanted columns by highlighting a column name or a group of names and clicking **Delete**.
- The name of a column can also be changed by right-clicking the column and choosing **Properties** to edit its properties.
- Any new name must conform to the database naming standards. Good practice is to use alphanumerics and the underscore character.
  Refer to **EDW 3NF Column Properties** for more details.

Tip:

When prototyping, and in the initial stages of an analysis area build, it is best not to remove columns, nor to change their names to any great extent. This type of activity is best left, and done after end users have used the data and provided feedback.

© Copyright, WhereScape Software Ltd All rights reserved.

Adding columns

With the columns of the EDW 3NF table displayed in the middle pane, this pane is considered a Drop Target for additional columns.

- Select the columns from other Load and/or Stage tables and drag these columns into the middle pane. The source table column in the middle pane shows where each column originated.
- The column **description** could be acquired from three different tables. Best practice is to rename at least two of the columns, perhaps also adding context to the column name. For example, description could become group_description, and so forth.
- There are a number of WhereScapeRED ancillary columns that do not have a source table. These columns have been added by WhereScape RED and are added depending on earlier choices. A description of these columns follows:

Column name	Description
dss_start_date	Used for history tables. This column provides a date time stamp when the EDW 3NF table record came into existence. It is used to ascertain which EDW 3NF table record should be used when multiple tables are available.
dss_end_date	Used for history tables. This column provides a date time stamp when the EDW 3NF table record ceased to be the current record. It is used to ascertain which EDW 3NF table record should be used when multiple tables are available.
dss_current_flag	Used for EDW 3NF history tables. This flag identifies the current record where multiple versions exist.
dss_source_system_key	Added to support history tables that cannot be fully conformed, and the inclusion of subsequent source systems. See the ancillary settings section for more details.
dss_version	Used for EDW 3NF history tables. This column contains the version number of an EDW 3NF history tables record. Numbered from 1 upwards with the highest number being the latest or current version. It forms part of the unique constraint for the business key of an EDW 3NF history tables.
dss_update_time	Indicates when the record was last updated in the data warehouse.
dss_create_time	Indicates when the record was first created in the data warehouse

Creating the table

Once the EDW 3NF table has been defined in the metadata, we need to physically create the table in the database.

1. This is done by right-clicking on the EDW 3NF table in the left pane and selecting **Create (ReCreate).**
2. The Results pane appears to show the results of the creation. The message confirms that the EDW 3NF table was created. A copy of the actual database create statement and if defined, the results of any index create statements is listed. For the initial create, no indexes will be defined.
3. If the table was not created, then ascertain and fix the problem. A common problem is a 'Duplicate column' where a column has the same name in two of the source tables. The best way to find such a column is to double-click the list heading **Col name**; which sorts the column names into alphabetical order.
4. Another double-click on the heading sorts the columns back into their create order.

© Copyright, WhereScape Software Ltd All rights reserved.

Generating the EDW 3NF Update Procedure

Once an EDW 3NF object has been defined in the metadata and created in the database, an update procedure can be generated to handle the joining of any tables and the update of the EDW 3NF object.

Notes

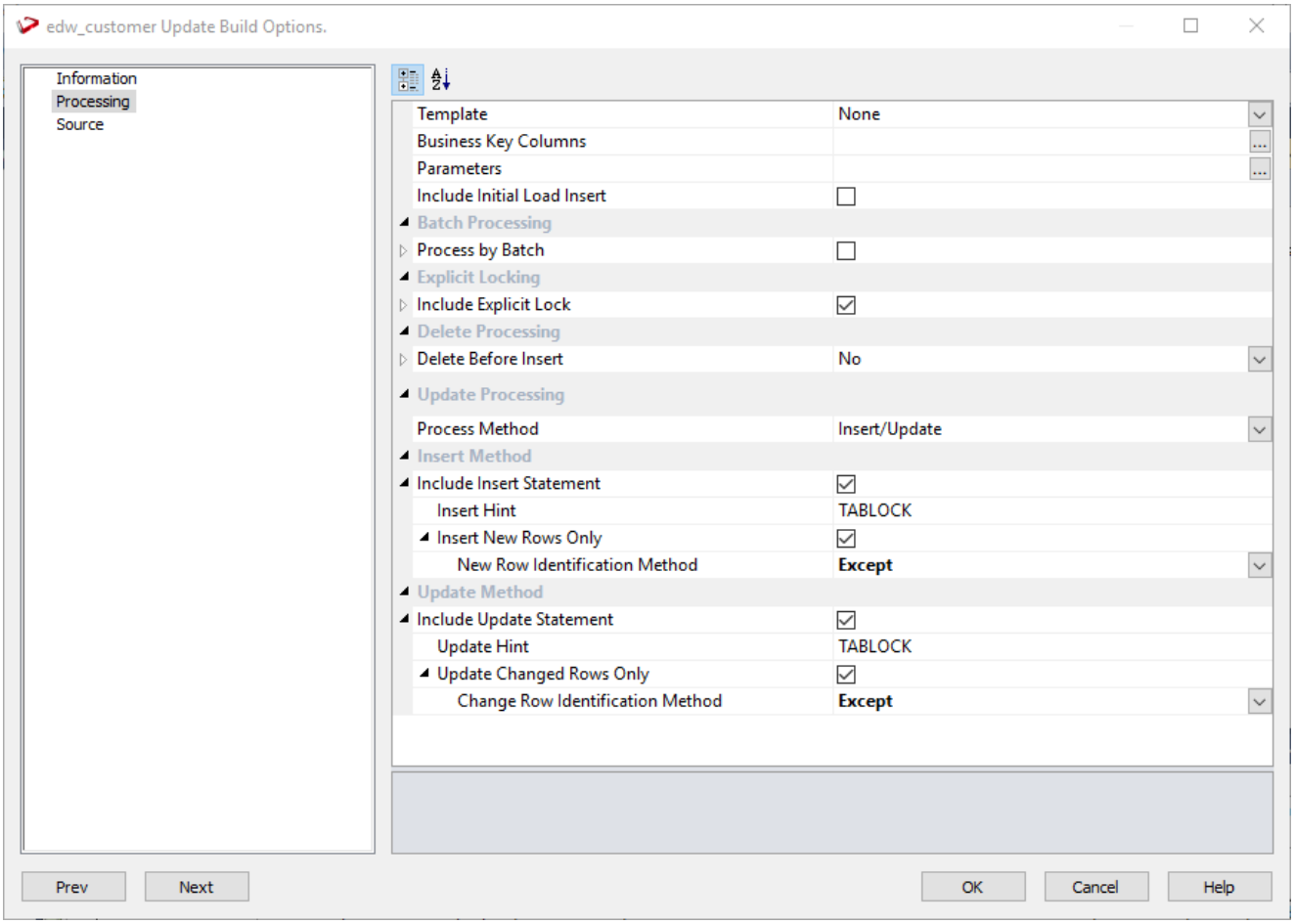
- You can also generate an update procedure via a template, refer to **Rebuilding Update Procedures** for details.
- If the EDW 3NF object is created in a custom database target, then an additional processing option is available—PowerShell script-based processing. Refer to **Script Templates for Custom Database Table Objects** for details.

Generating a Procedure

To generate a procedure:

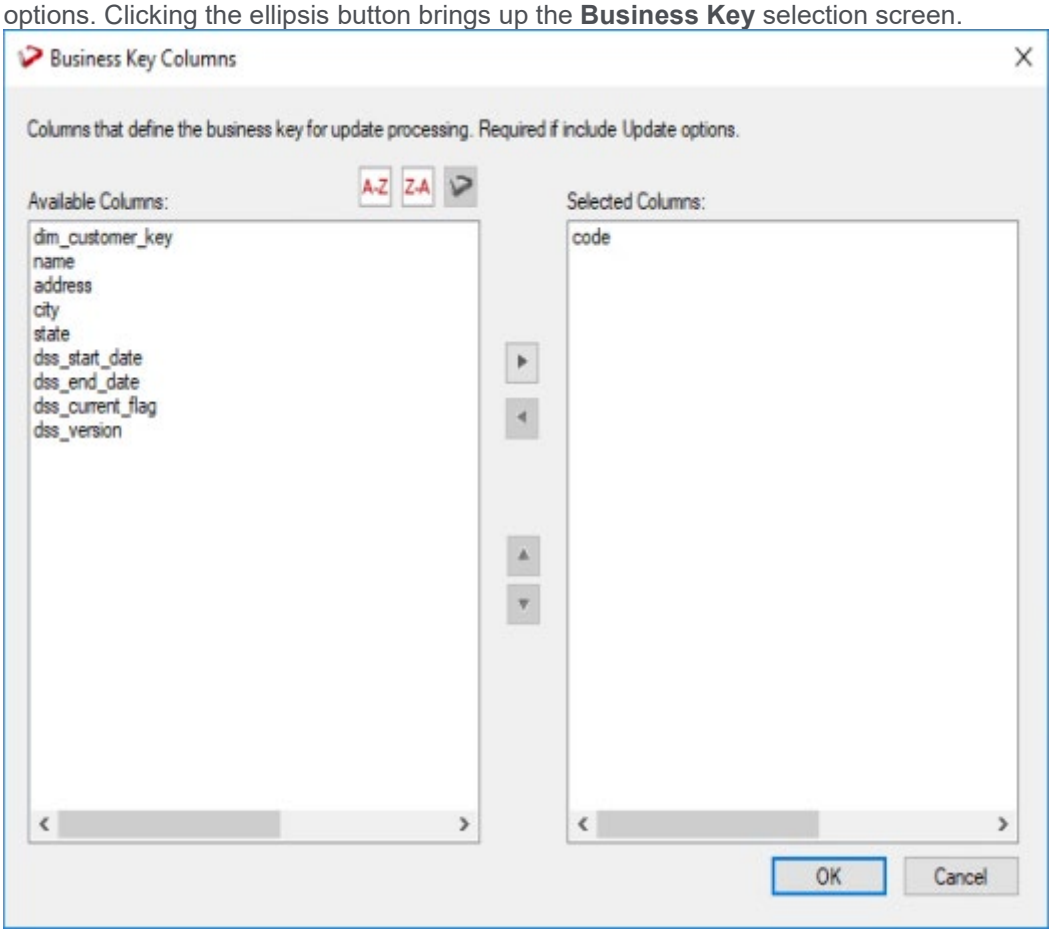
1. Right-click the EDW 3NF object in the left pane and select **Properties**.
2. Click the **Rebuild** button to start the process of generating the new procedure.
3. A series of options are available.

Processing Tab

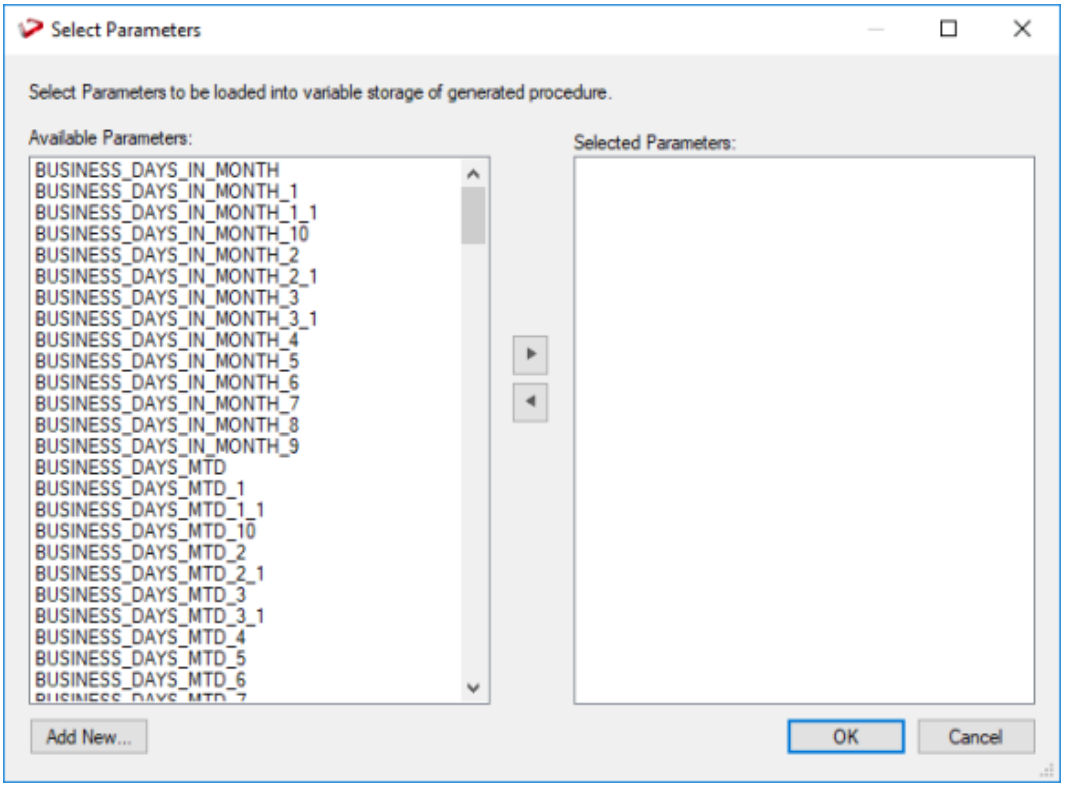
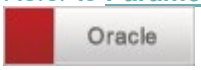


Options	Description
Template	Enables you to generate update procedures via a **template**.
Business Key	Columns that define the business key for update processing. Required for include Update

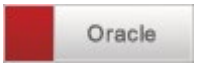
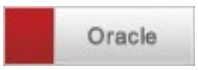
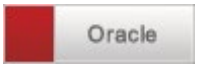
© Copyright, WhereScape Software Ltd All rights reserved.

Options	Description
Columns	options. Clicking the ellipsis button brings up the **Business Key** selection screen.  A business key can be made up of multiple columns, but it must provide a unique identifier. Where multiple columns separately uniquely identify rows in the EDW 3NF Object, choose one to act as the primary business key. For example, a source table may have a unique constraint on both a product code and a product description. Therefore the description as well as the code must be unique. It is possible to combine the two columns, but the normal practice would be to choose the code as the business key. **Tips** • Use the column name ascending/descending buttons to sort column names. To revert to the meta column order, click on the meta column order button. • **NULL Values** - none of the columns chosen as the business key should ever contain a NULL value.
Parameters	Any parameters selected are included in the generated update procedure as variables. The procedure includes code to retrieve the value of the parameter at run time and store it in the declared variable. Clicking the ellipsis button brings up the **Parameters** selection screen.

© Copyright, WhereScape Software Ltd All rights reserved.

Options	Description
	 The variables can also be used in column transformations and in the from/where clause for the update procedure. Some databases have a 30 character limit for variable names. WhereScape RED ensures the variables added for any parameters are less than 30 characters long by creating variable names in the form v_ followed by the first 28 characters of the parameter name. For example, a parameter called MINIMUM_ORDER_NUMBER_SINCE_LAST_SOURCE_LOAD will be available as the variable v_MINIMUM_ORDER_NUMBER_SINCE_L. **Tips** • WhereScape RED parameters should be unique within the first 28 characters to avoid conflicting variables names. • If the desired parameter doesn't exist in the metadata yet, a new parameter can be added by clicking on the Add New button on the bottom leftmost corner of the Select Parameters dialog. • Refer to **Parameters** for details. •  **Enable Parallel DML** - adds all the code required to the update procedure to enable Oracle parallel inserts. Default for this option is not set.
Include Initial Load Insert	Adds an additional insert statement to the update procedure that runs if the target EDW 3NF object is empty. The benefit of this is improved performance inserting into an empty table without performing any checks to see if rows already exist. The default for this field is off (i.e. an initial insert statement is not added to the procedure).
Process by Batch	Enables you to select a column to drive data processing in a loop based on the distinct ordered values of the selected columns. The update procedure loops on this column and performs the delete, update and/or insert for each value. If the column chosen is a date data type (date, datetime or timestamp), then the user is able to specify yearly, monthly, daily or column level looping. The default for this field is off (do not do batch processing).

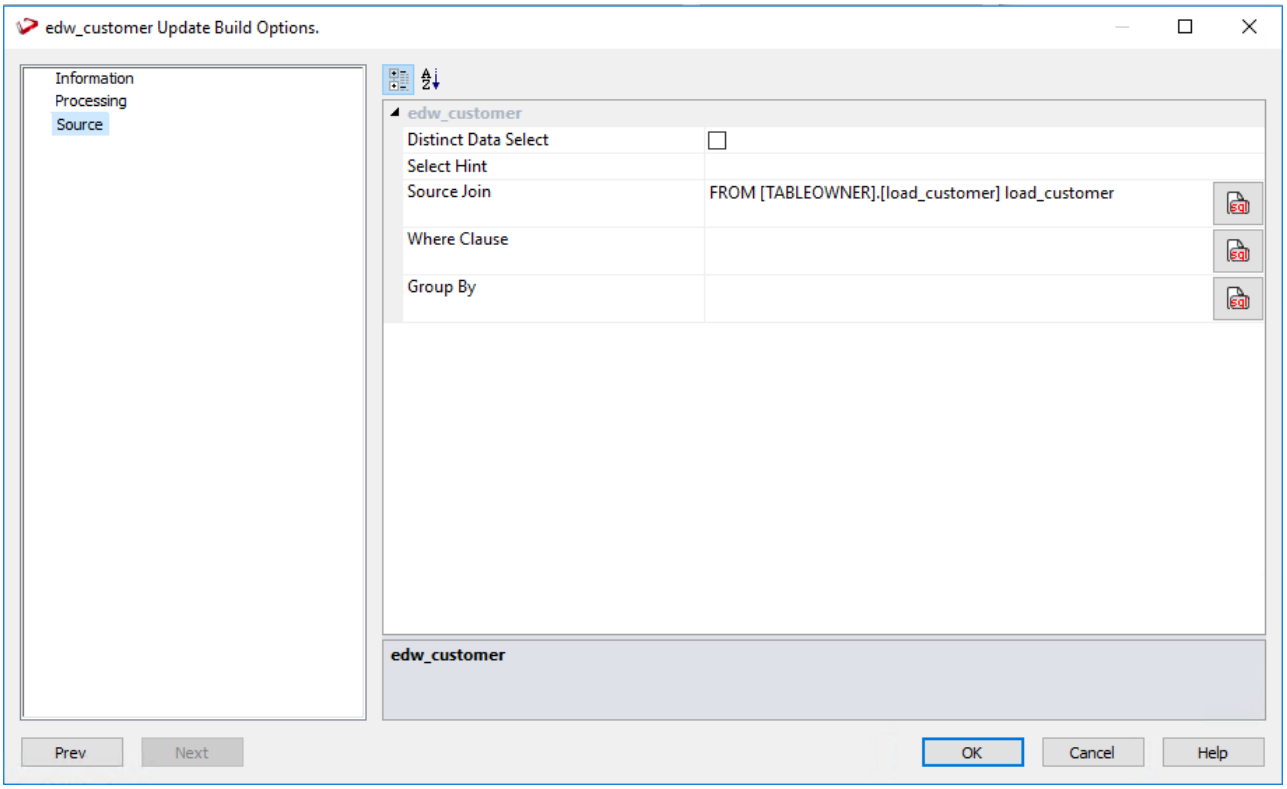
© Copyright, WhereScape Software Ltd All rights reserved.

Options	Description
Delete Before Insert	Enables you to select how to process deletes. It enables a delete statement to be added to the update procedure before any update or insert statement. This is a particularly useful option for purging old data and for updates based on a source system batch number. If this option is selected, the following options are also available:  **Truncate Option** – optional Oracle TRUNCATE clause which is appended to the truncate statement. Add a truncate option such as REUSE/DROP STORAGE by typing it in the truncate options box. • **Issue Warning if a Delete occurs:** this option sets the procedure to a warning state if any deletes occur. • **Delete Where Clause:** the delete where clause is appended to the generated delete statement to constrain the rows deleted.
Process Method	Select between Insert/Update and Merge which allows you to use the Merge statement instead of two separate Insert and update statements.
Include Insert Statement	Includes the insert statement in the procedure to insert new rows in the EDW 3NF Object. **Tip: Insert** Enter a database compliant hint to be used in the INSERT statement. This is an Oracle and SQL Server only option. Defaults can be configured in **Tools > Options > Default Update Procedure Options**. Default is TABLOCK.  Default is APPEND.
Insert New Rows Only	Uses change detection to work out what rows require inserting.
New Row Identification Method	Method used to identify that records in source are not currently recorded in the target table.
Existing Data Selection Hint	Database-compliant hint to be used for the existing data select statement.
Include Update Statement	Includes an update statement in the procedure to update changing rows in the EDW 3NF Object. If this option is chosen, then the **Update Changed rows** only option is available. **Tip: Update** Enter a database compliant hint to be used in the UPDATE statement. This is an Oracle and SQL Server only option. Defaults can be configured in **Tools > Options > Default Update Procedure Options**. Default is TABLOCK.  Default is APPEND.
Update Changed Rows Only	Uses change detection to work out what rows require updating. Choosing this option, enables the **Change Row identification Method**.
Change Row Identification Method	Method used to identify that records in source have changed from what is currently recorded in the target table.

© Copyright, WhereScape Software Ltd All rights reserved.

Options	Description
Existing Data Selection Hint	Database-compliant hint to be used for the existing data select statement. **Tip: Merge** Enter a database hint to be used in the MERGE statement. This is an Oracle and SQL Server only option. Defaults can be configured in **Tools > Options > Default Update Procedure Options**. SQL Server — Default is TABLOCK. Oracle — Default is APPEND.

Source Tab



Options	Description
Distinct Data Select	Ensures duplicate rows are not added to the EDW 3NF Object. This is achieved by the word DISTINCT being added to the source select in the update procedure. This not selected by default. Select Hint: A database compliant hint to be used in the source Select statement.
Source Join	The From clause, including Source Join information.
Where Clause	The Where clause.
Group By	The Group By clause.

Simple Join

A simple join only returns rows where data is matched in both tables. So for example if table A has 100 rows and table B has a subset of 24 rows. If all the rows in table B can be joined to table A then 24 rows will be returned. The other 76 rows from table A will not be returned.

© Copyright, WhereScape Software Ltd All rights reserved.

Outer Join

The outer join returns all rows in the master table regardless of whether or not they are found in the second table. So if the example above was executed with table A as the master table then 100 rows would be returned. 76 of those rows would have null values for the table B columns.

Notes

- When WhereScape RED builds up an outer join, it needs to know which table is the master table and which is subordinate. Select the join column from the master table first. In the example screen above the table 'load_order_header' has had its column chosen and the column for the table 'load_order_line' is currently being chosen. This will result in the 'load_order_header' table being defined as the master, as per the example statement above. The results of this example select are that a row is added containing order information, regardless of whether or not a corresponding load_order_line entry exists.
- When upgrading from a RED version previous to 6.8.2.0 and moving existing objects to a target location, all procedures that reference those objects will need to be rebuilt.
  Any **FROM** clauses will also need to be manually regenerated in order for the table references to be updated to the new [TABLEOWNER] form.

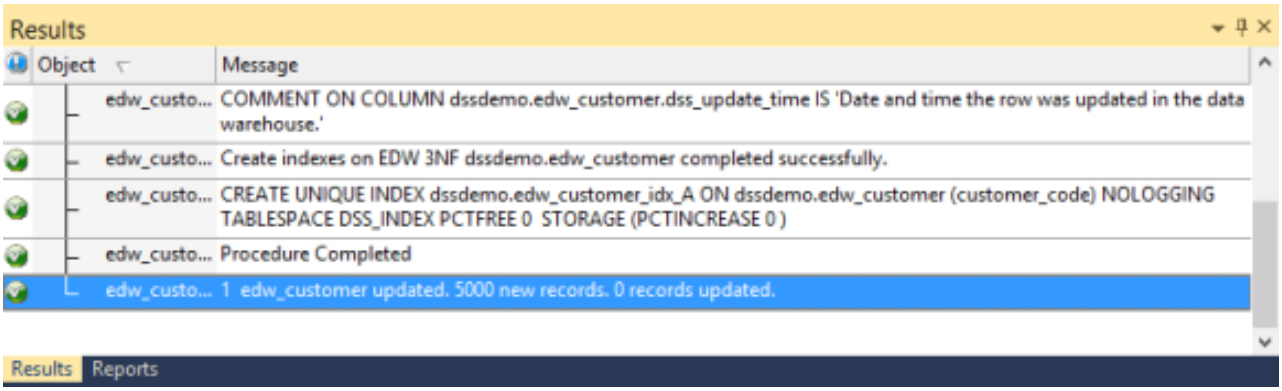
Building and Compiling the Procedure

- Once the relevant options are completed, click **OK**. The procedure is built and compiled.
- If the compile fails an error is displayed along with the first few lines of error messages. Compile fails typically occur when the physical creation of the table was not done.
- If the compile fails for some other reason the best approach is to use the procedure editor to edit and compile the procedure. The procedure editor highlights all the errors within the context of the procedure.
- Once the procedure has been successfully compiled, it can either be executed interactively or passed to the scheduler.

© Copyright, WhereScape Software Ltd All rights reserved.

Indexes

By default, a number of indexes will be created to support each EDW 3NF table. These indexes are added once the procedure has been built.
An example of the type of indexes created is as follows:



Additional indexes can be added, or these indexes changed. Refer to **Indexes** for more details.

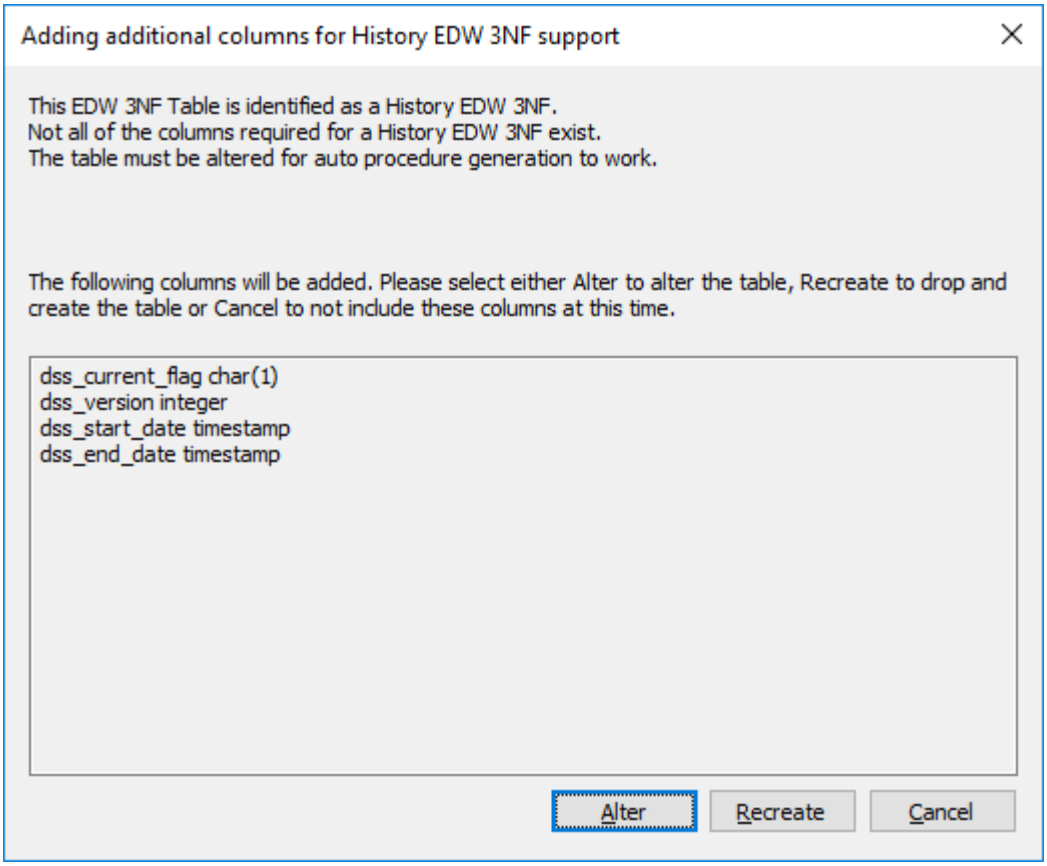
Converting an existing EDW 3NF Table to an EDW 3NF History Table

To convert an EDW 3NF table to an EDW 3NF history table:

- Change the table type to **History** in the Properties screen.
- Select **(Build Procedure...)** from the **Update Procedure** drop-down list and click **OK**.

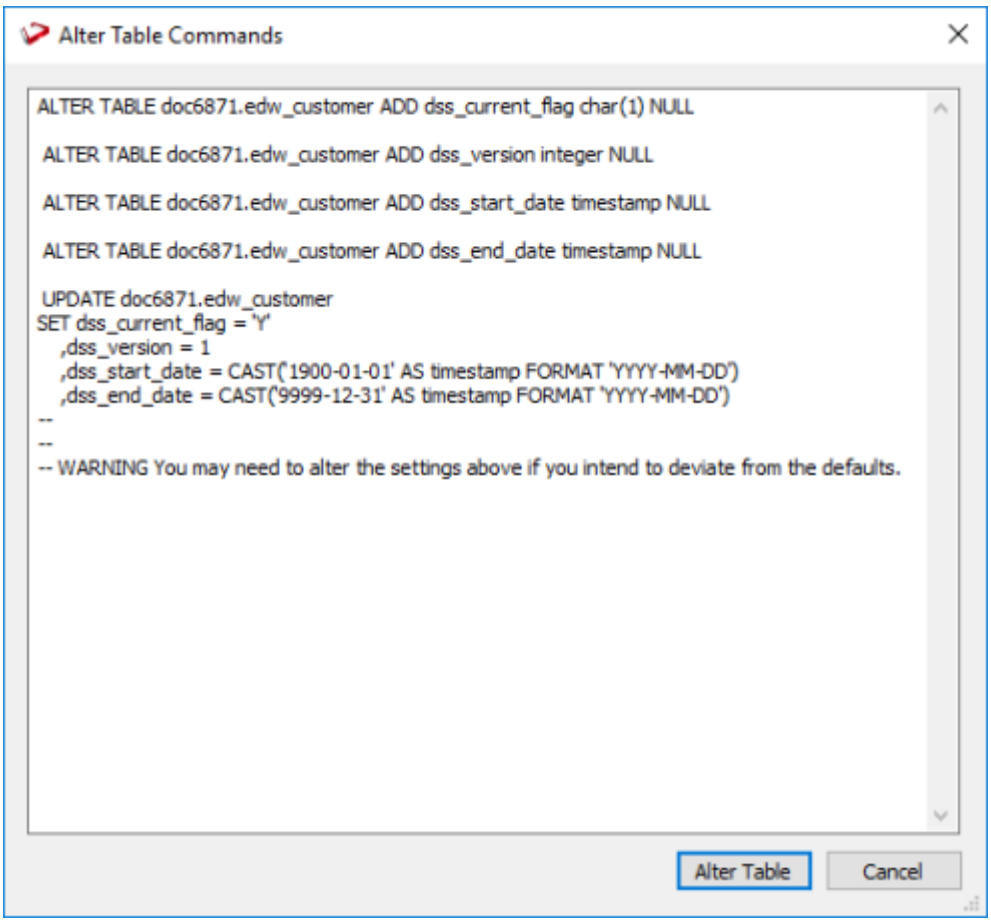
If the existing EDW 3NF table is **NOT** to be dropped and recreated, then the following process must be followed:

1. Click **OK** in the **Procedure Type** window.
2. Click **OK** in the **Business Key** window.
3. Click **Alter** in the **Adding Additional Columns** window.



© Copyright, WhereScape Software Ltd All rights reserved.

4. Click **Alter Table** on the **Alter Table Commands** window:



Note

The SQL statement in this window can be edited if required.

5. Click **OK** on all remaining windows.

EDW 3NF Table Artificial Keys

By default, EDW 3NF tables in WhereScape RED do not have an artificial (surrogate) key. Artificial keys can be added manually.

The quickest way to do this is to add an extra column to the EDW 3NF table by using either **Add Column** or **Copy Column**.

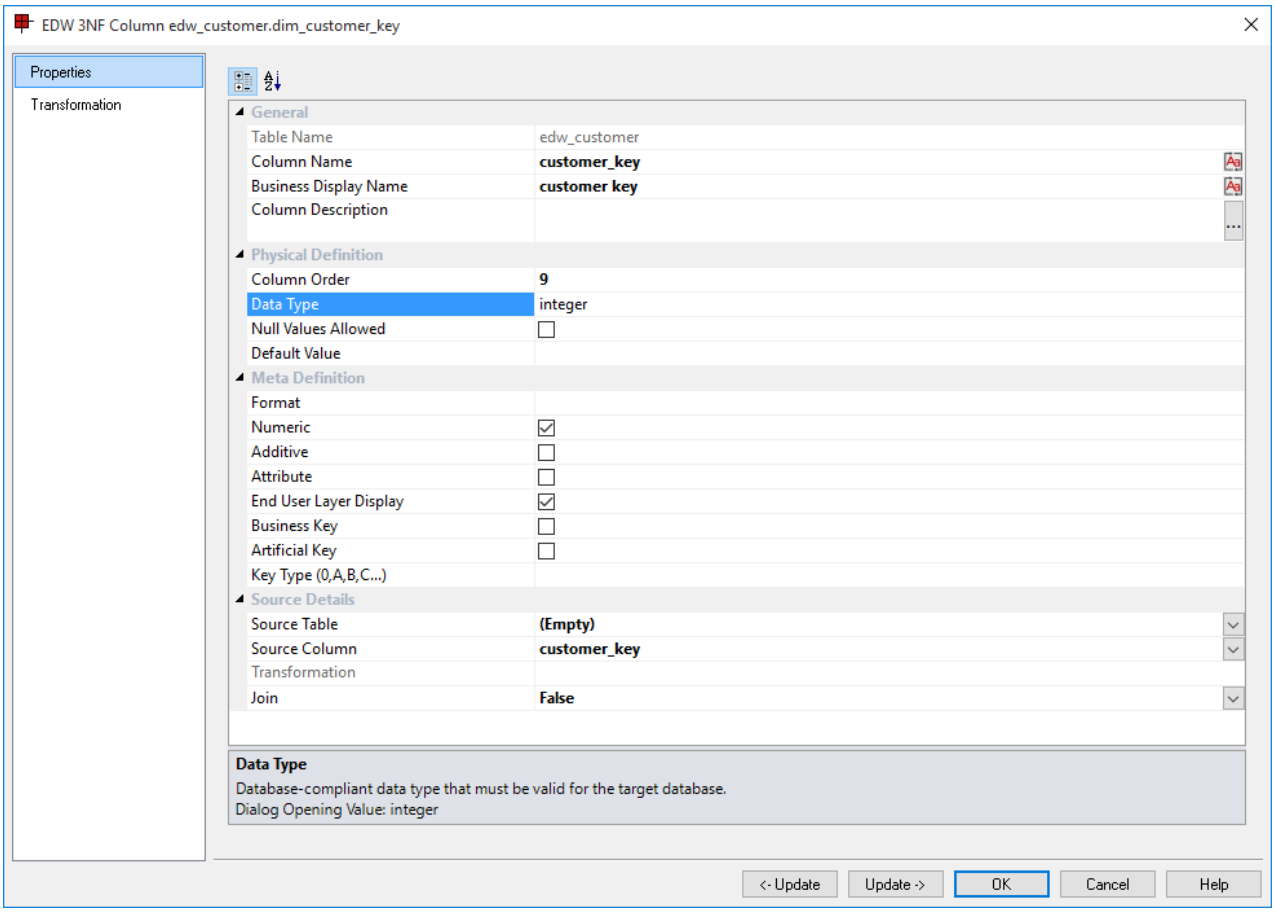
Edit the properties of the new column to have the correct name and order, source table and column, datatype, key type and flags.

Specifically:

- The **Column Name** and **Source Column** should be the same.
- The **Source Table** should be empty.
- The **Data Type** should be:
- DB2: integer generated by default as identity (start with 1, increment by 1)
- Oracle: integer
- SQL Server: integer identity(0,1)
- The **Key Type** should be 0.
- Only the **Numeric** and **Artificial Key** flags must be set.

© Copyright, WhereScape Software Ltd All rights reserved.

The following example shows a manually added artificial key column:



The artificial key for an EDW 3NF table is set via a sequence in Oracle and an identity column in SQL Server and DB2. This artificial key normally, and by default, starts at one and progresses as far as is required.

A WhereScape standard for the creation of special rows in the EDW 3NF tables is as follows:

Key value	Usage
1 upwards	The standard artificial keys are numbered from 1 upwards, with a new number assigned for each distinct EDW 3NF table record.
0	Used as a join to the EDW 3NF table when no valid join existed. It is the convention in the WhereScape generated code that any EDW 3NF table business key that either does not exist or does not match is assigned to key 0.
-1 through -9	Used for special cases. The most common being where an EDW 3NF table is not appropriate for the record. A new key is used rather than 0 as we want to distinguish between records that are invalid and not appropriate.
-10 backward	Pseudo records. In many cases, we have to deal with different granularities in our data. For example, we may have a table that contains actual sales at a product SKU level and budget information at a product group level. The product table only contains SKU based information. To be able to map the budget records to the same table, we need to create these pseudo keys that relate to product groups. The values -10 and backwards are normally used for such keys.

Note

To have a surrogate key auto added for EDW 3NF tables, refer to **Global Naming of Key Columns** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

EDW 3NF Table Column Properties

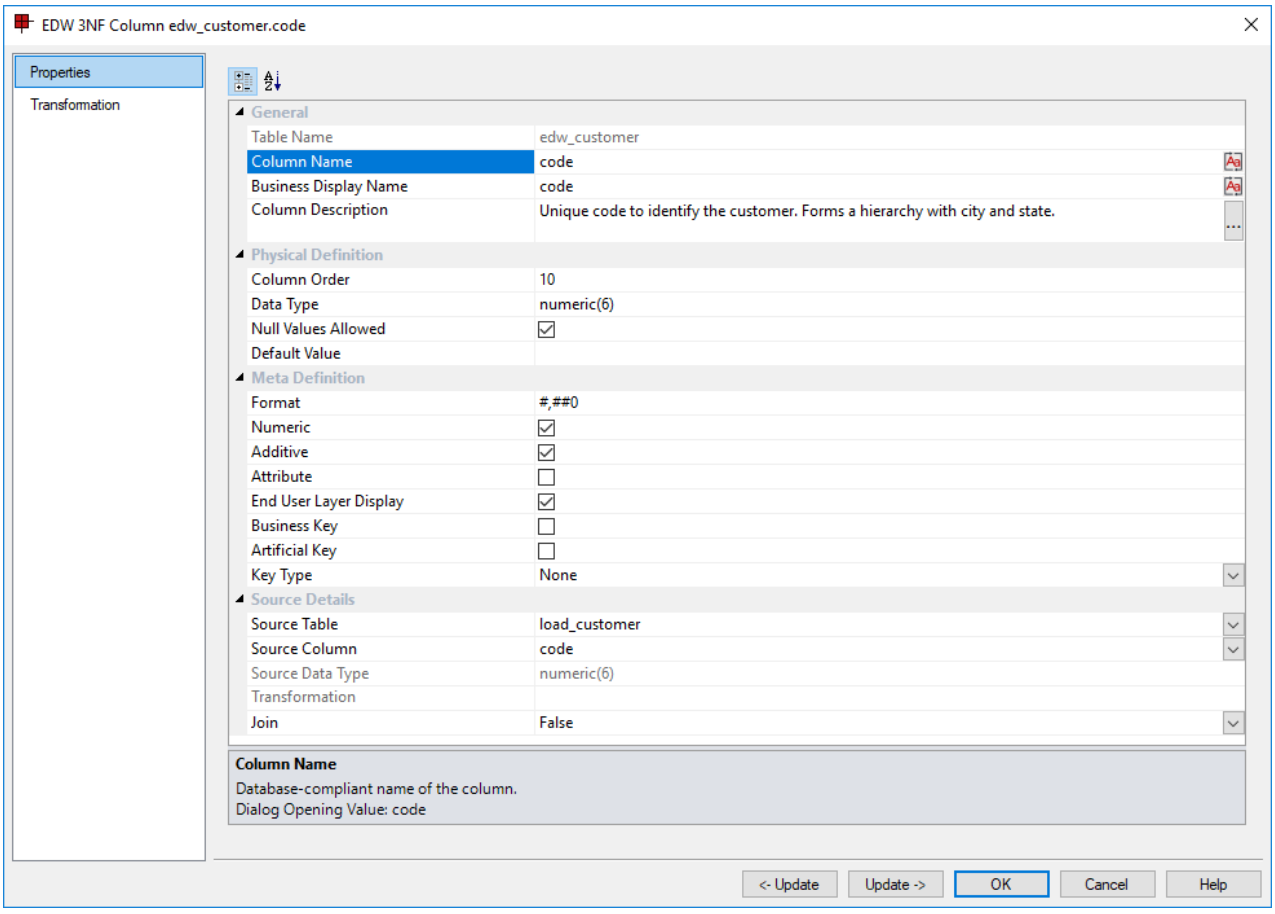
Each EDW 3NF table column has a set of associated properties. The definition of each property is described below:

- If the **Column name** or **Data type** is changed for a column then the metadata will differ from the table as recorded in the database.
- Use the **Validate > Validate Table Create Status** menu option or the right-click menu to compare the metadata to the table in the database.
- A right-click menu option of **Alter table** is available when positioned on the table name after the validate has completed. This option will alter the database table to match the metadata definition.

Tip:

If a database table's definition is changed in the metadata then the table will need to be altered in the database.
Use the **Validate > Validate Table Create Status** to compare metadata definitions to physical database tables. The option also provides the ability to alter the database table, through a pop-up menu option from the validated table name.

Sample EDW 3NF **Properties** screen:



Tip:

The two special update keys allow you to update the column and step either forward or backward to the next column's properties.
ALT-Left Arrow and **ALT-Right Arrow** can also be used instead of the two special update keys.

© Copyright, WhereScape Software Ltd All rights reserved.

General

Field	Description
Table Name	Database-compliant name of the table that contains the column. [Read-only].
Column Name	Database-compliant name of the column. Typically column-naming standards exclude the use of spaces etc. A good practice is to only use alphanumerics, and the underscore character. Changing this field alters the table's definition. **Note:** A case conversion button on the right converts the text between different cases: UPPER CASE, Capitalized Case and lower case. The mode cycles to the next case in the sequence each conversion.
Business Display Name / Column Title	Name that the business uses to refer to the column, which is included in the RED-generated documentation and can be used in the end user layer of other tools. [Does NOT affect the physical database table]. As such it is a free form entry and any characters are valid. **Note** A case conversion button on the right converts the text between different cases: UPPER CASE, Capitalized Case and lower case. The mode cycles to the next case in the sequence each conversion.
Column Description	This field contains the description for the column. It may be a description from a business user's point of view. This field might additionally contain information on where and how the column was acquired. For example if the column is sourced from multiple tables or is a composite or derived column then this definition would normally describe the process used to populate the column. This field is used in the documentation and is available via the view **ws_admin_v_dim_col**. This field is also stored as a comment against the column in the database.

Physical Definition

Field	Description
Column Order	Numeric value that controls the relative order of columns in the database create statement. The lowest numbered column will appear first in the table. Although this affects the physical table definition no action will be taken unless the table is re-created in the database. The columns can be re-numbered based on the existing order by choosing the **Respace Order Number** pop-up menu option when positioned over any column in the table. This action will number the columns in increments of 10 starting at 10. In addition to a simple change of the order field, the column order can be changed by first displaying the columns in the middle pane and then using drag and drop to move the columns around. This drag and drop process will automatically renumber the columns as required
Data Type	Database-compliant data type that must be valid for the target database. Typical data types for Oracle are integer, number, char(), varchar2() and date. For SQL Server common types are integer, numeric, varchar() and datetime. See the database documentation for a description of the data types available. Changing this field alters the table's definition.
Null Values Allowed	Determines whether the table column can hold NULL values or whether a value is always mandatory.
Default Value	Initial value that is assigned to the column when a row is inserted into the table but no value is specified for the column.

© Copyright, WhereScape Software Ltd All rights reserved.

Meta Definition

Field	Description
Format	Optional format mask that can be used in end user tools. [Does NOT affect the physical database table]. As such it is a free form entry and any characters are valid. Typically format masks are only used on numeric fields. Example: #,###0.00. It is not worth the effort of populating this field unless it can be utilized by the end user tools in use.
Numeric	Indicates whether the table column holds values that are numeric. This is normally only relevant for fact tables. It does not affect the physical table definition, but rather provides input to the view **ws_admin_v_dim_col** which can be used to assist in the population of an end user tool's end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.
Additive	Indicates whether the table column holds values that are additive. This implies that the column can be summed when performing data grouping in a query. This is normally only relevant for fact tables. It does not affect the physical table definition, but rather provides input to the view **ws_admin_v_dim_col** which can be used to assist in the population of an end user tool's end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.
Attribute	Indicates whether the table column holds values that are descriptive, and/or are used for grouping/summing. An attribute is defined as a column that is non factual. For example we may have an order number, or an invoice number stored in the fact table. Such columns are considered attributes, rather than facts. This checkbox is therefore normally only relevant for fact tables. This checkbox does not affect the physical table definition, but rather provides input to the view **ws_admin_v_dim_col** which can be used to assist in the population of an end user tools end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.
End User Layer Display	Indicates whether the table column is available/visible to end users. If set the documentation will include the column in the glossary and in the user documentation. It is also used to decide what columns appear in the view **ws_admin_v_dim_col.** Typically columns such as the artificial key would not be enabled for end user display
Business Key	Indicates whether the column is part of the BUSINESS key, which is defined during the update procedure build. [Normally maintained automatically]. Multiple columns can form the primary business key.
Artificial Key	Indicates whether the column is the artificial key. Only one artificial key is supported. This indicator is set by WhereScape RED during the initial drag and drop creation of a table, and should not normally be altered.
Key Type	Key type that is assigned and used when generating the table's update procedure and indexes. [Normally maintained automatically]. It can be altered here, but this should only be done if the consequences are fully understood and tested. The supported values are:

Key type	Meaning
0	The artificial key. Set when the key is added during drag and drop table generation.
1	Component of all business keys. Indicates that this column is used as part of any business key.
A	Indicates that the column is part of the primary business key. Set whenever a business key is defined as part of an update procedure generation.
B-Z	Indicates that the column is part of a secondary business key. Only used during index generation and not normally set.

© Copyright, WhereScape Software Ltd All rights reserved.

Source Details

Field	Description
Source Table	Identifies the source table where the column's data comes from. This source table is normally a load table within the data warehouse. If the column was sourced from multiple tables, then the normal practice is to record one of the tables in this field and a comment listing all of the other tables in the **Source Strategy** field. This field is used when generating a procedure to update the EDW 3NF table. It is also used in the track back diagrams and in the documentation.
Source Column	Identifies the source column where the column's data comes from. Such a column is normally a load table column, which in turn may have been a transformation or the combination of multiple columns. For previous value managed columns the source column is the column in the table whose previous value is being recorded.
Source Data Type	Identifies the source column's data type. [Read-only].
Transformation	Refer to **EDW 3NF Table Column Transformations**. [Read-only].
Join	Indicates whether the table column is used in a table join. [Normally maintained automatically but can be optionally changed to override the default join logic used in the generated update procedure]. The **Source Table** and **Source Column** fields will provide the other EDW 3NF table's side of the join. The options for this field are: False, True, Manual and Pre Join. • Setting this field to Manual changes the way the dimension table is looked up during the update procedure build. It allows you to join the dimension manually in the **Source Join** wizard (used to build the 'Where' clause). • Setting this field to Pre Join activates the **Pre Join Source Table** field and allows you to select a table from the drop-down list.
Pre Join Source Table	Indicates the table from which the pre joined column was sourced. When the Join option is set to False, this field becomes inactive. When the Join option is set to True or Manual, this field is set to the current table name. When the Join option is set to Pre Join, then you can select the required table from the drop-down list.

© Copyright, WhereScape Software Ltd All rights reserved.

EDW 3NF Table Column Transformations

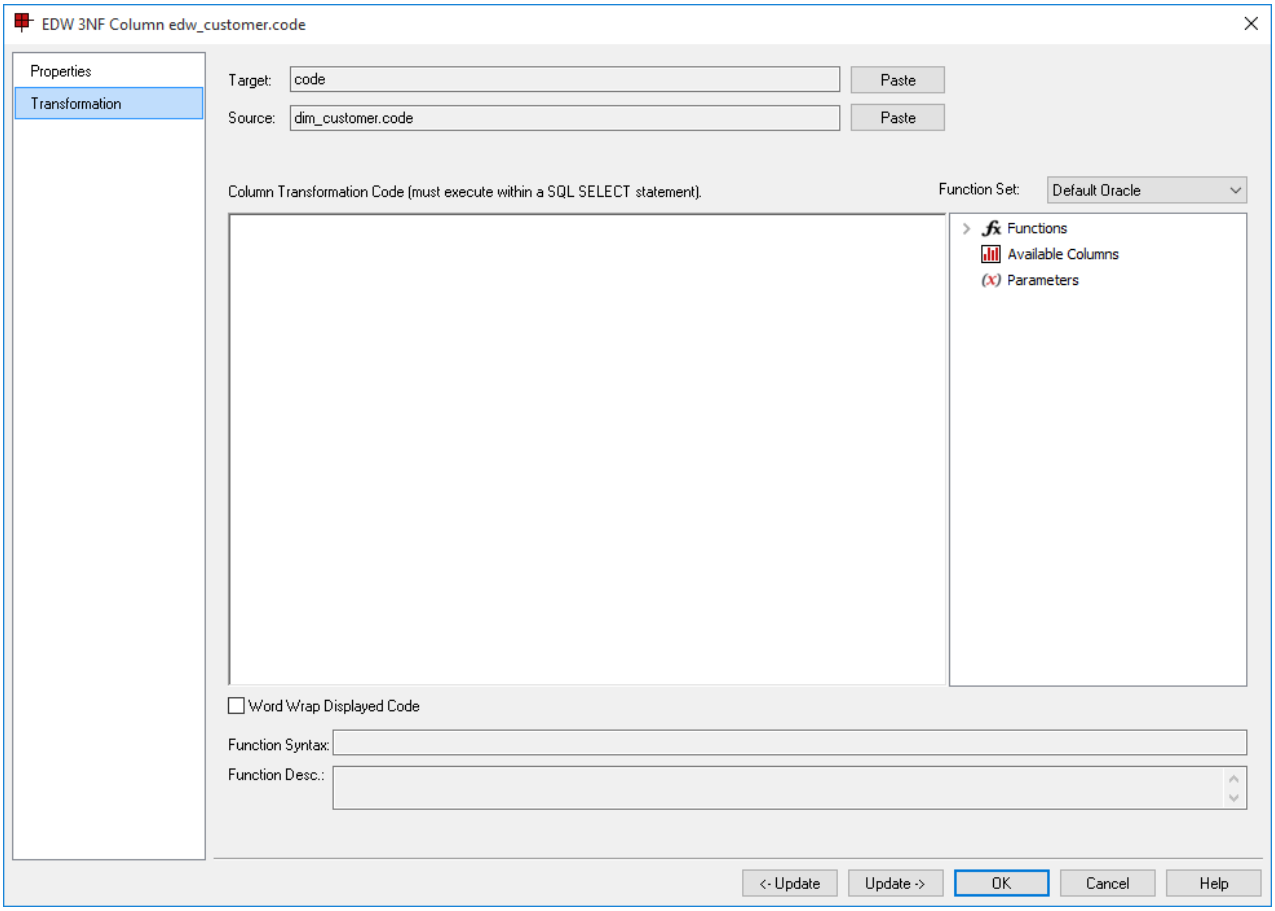
Each EDW 3NF table column can have a transformation associated with it. The transformation will be included in the generated procedure and will be executed as part of the procedure update.

The transformation must therefore be a valid SQL construct that can be included in a **Select** statement.

For example we could have a transformation of 'load_order_line.quantity * 0.125' to calculate a tax column of 12.5%.

Click the **Transformation** tab to enter a transformation.

The transformation screen is as follows:



Note:

Transformations are only put into effect when the procedure is re-generated.

Tip:

Microsoft Analysis Services 2005+ Tabular Mode Tables: For Tabular Mode table column transformations, **Default DAX** is the only applicable Function Set for **after load** transformations.

Refer to **Transformations** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

Data Vaults

The Data Vault system is an alternative approach to modeling an enterprise data warehouse that has been gaining popularity among organizations.

The Data Vault data warehouse architecture was invented by Dan Linstedt to provide an alternative to the traditional data warehouse modeling approach that includes developing 3rd Normal Form (3NF) type models or dimensional star schema models. The data vault methodology seeks to improve the efficiency of data ingestion and the flexibility of structure changes. For more information about Data Vaults, please refer to Dan Linstedt's website (http://danlinstedt.com/solutions-2/data-vault-basics/).

WhereScape RED has been enhanced to expand its current Data Vault functionality and provide improved automation for creating and managing Data Vault objects in WhereScape RED managed Data Warehouses. This includes the following:

- DSS columns for Load tables
- Wizard for Hash key generation
- Wizard for building Hub, Link and Satellite tables
- Templates for Procedure generation

All these enhancements are designed to be compliant with Data Vault 2.0 standards and are described in the succeeding sections below.

Note that previous releases of WhereScape RED used a workflow for Data Vault objects that is similar to the workflow for creating Data Store Objects. The Hub, Link and Satellite tables are based on standard Load or Stage tables (that do not include the hash key column type flags), WhereScape RED reverts to this behavior and the resulting procedures are generated by internal WhereScape RED automation and not via templates.

If the above legacy method of creating Data Vault Objects is required, please refer to the Data Store Objects chapter of this User Guide. You can also refer to the older versions of this User Guide, for details on the previous processes used to import Data Vault objects into WhereScape RED.

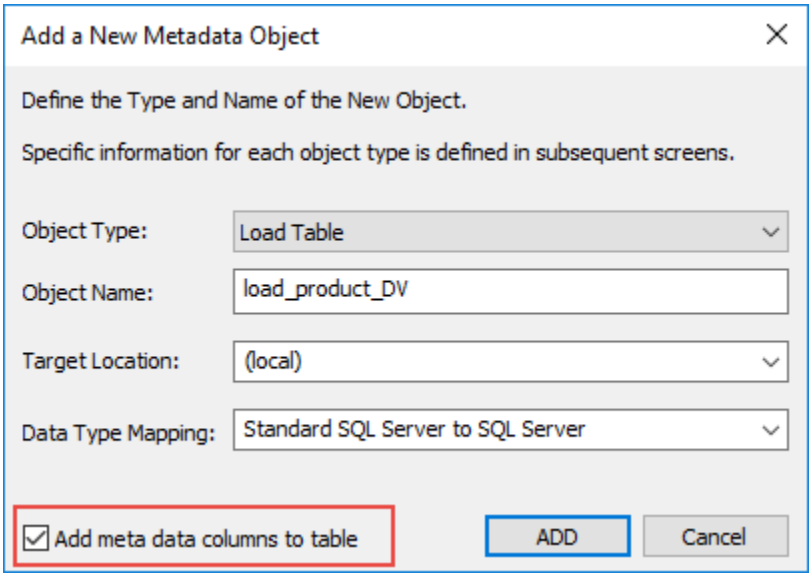
For more information on Data Vault design, refer to *Building a Scalable Data Warehouse With Data Vault 2.0* by *Daniel Linstedt and Michael Olschimke*.

Data Vault Functions and Features

The following describe the WhereScape RED functions and features that are used for building Data Vault objects (Hub, Link and Satellite) to create a Data Vault model.

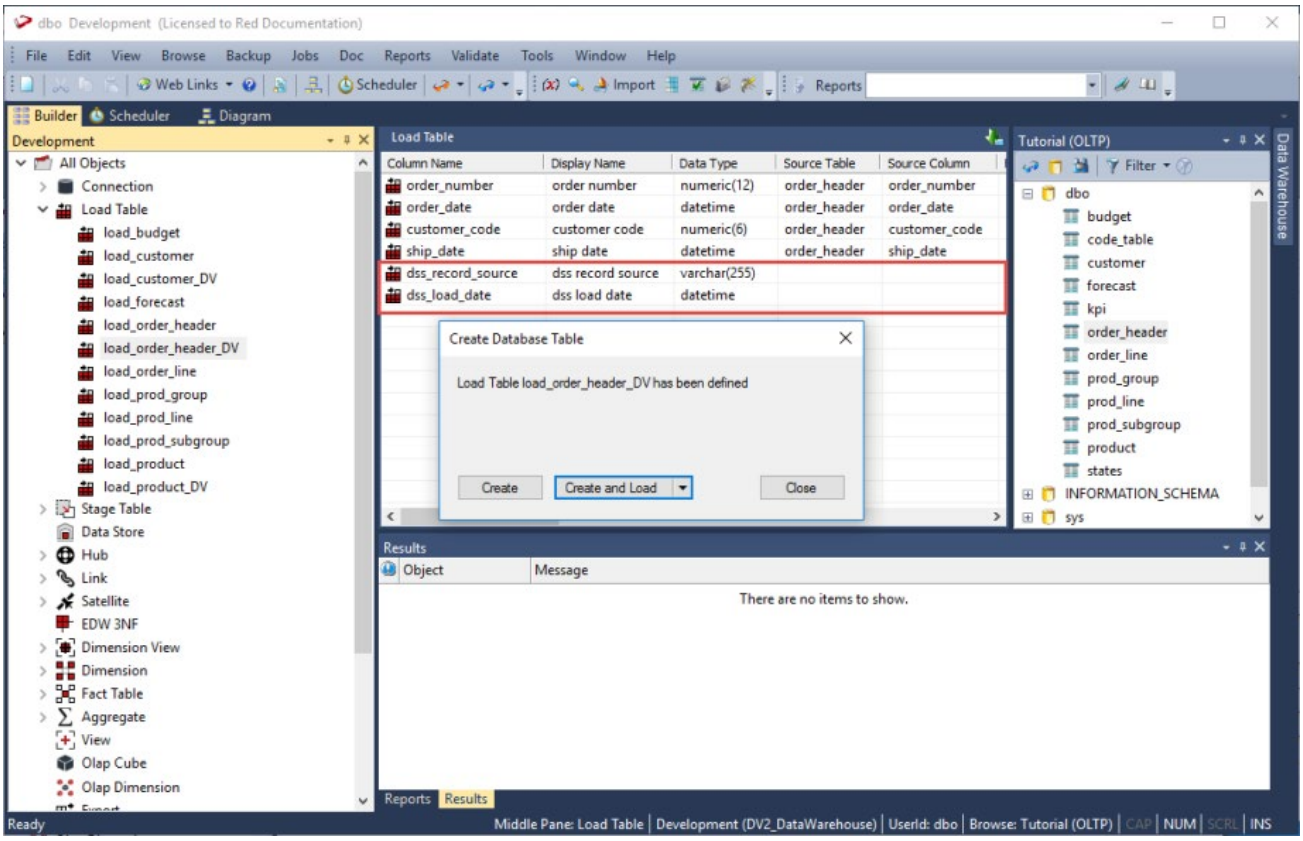
Load Table Meta Data Columns

The option to add default meta data columns to a new Load table object in WhereScape RED.



© Copyright, WhereScape Software Ltd All rights reserved.

If the option **Add meta data columns to table** is selected, two DSS columns (**dss_record_source** and **dss_load_date**) are included in the meta data for the table and are populated by transformations. Note that these two DSS columns could equally be applied to other Load tables not used in a Data Vault model but are particularly important to comply with the Data Vault standards.



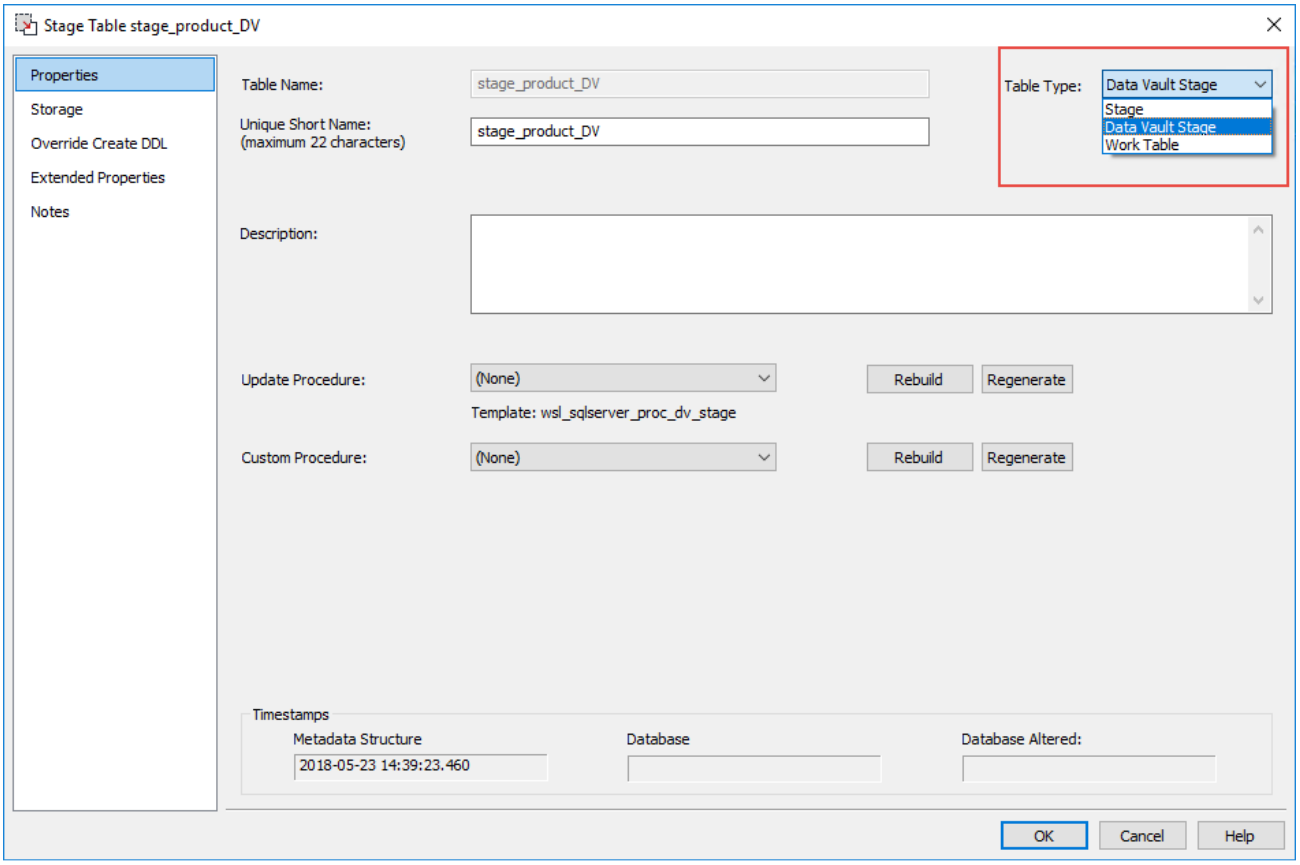
Note

Custom aliases for the **dss_load_date** and **dss_record_source** columns can be defined in the **Tools > Options > DSS Tables and Columns > DSS Columns to Include and Naming** window. If custom DSS aliases are defined then the aliases are used for the column names that are added to the new load tables. Refer to **DSS Columns** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

Data Vault Stage Table

WhereScape RED provides the option to create **Data Vault Stage** objects that are used to define the source columns for the Hub, Link and Change hash key columns.

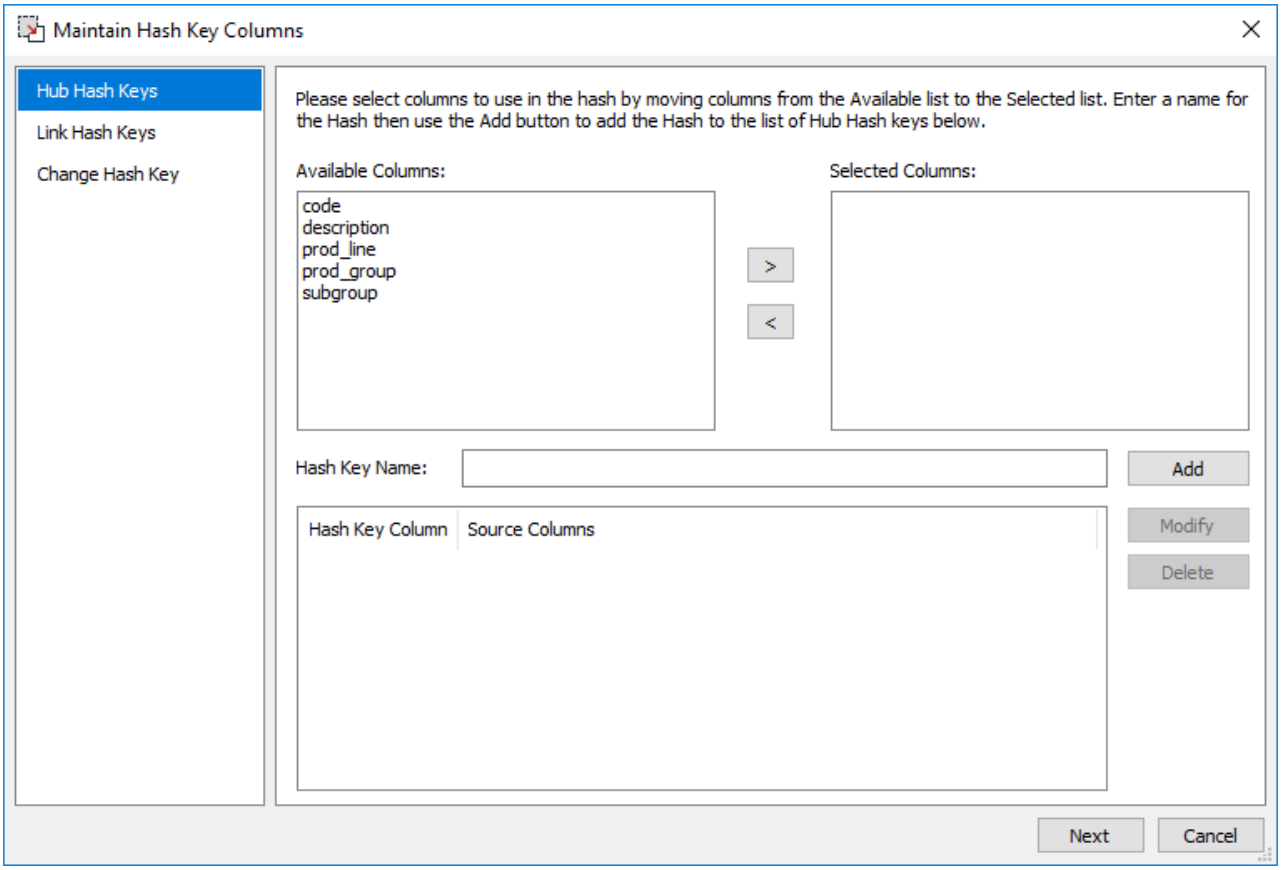


The **Data Vault Stage** option can be selected from **Table Type** drop-down of the **Stage Table Properties** screen. This object type is created via a Wizard, described in **Creating Data Vault Stage Tables**.

© Copyright, WhereScape Software Ltd All rights reserved.

Hash Key Generation Wizard

This Wizard is launched when building a **Stage** table with **Table Type** of **Data Vault Stage**. It enables you to specify the source columns to be used in defining the **Hub**, **Link** and **Change** hash key columns.



The generated Hash Keys are used to build the **Hub**, **Link** and **Satellite** objects in WhereScape RED. The detailed steps for using this Wizard is described in **Creating Data Vault Stage Tables**.

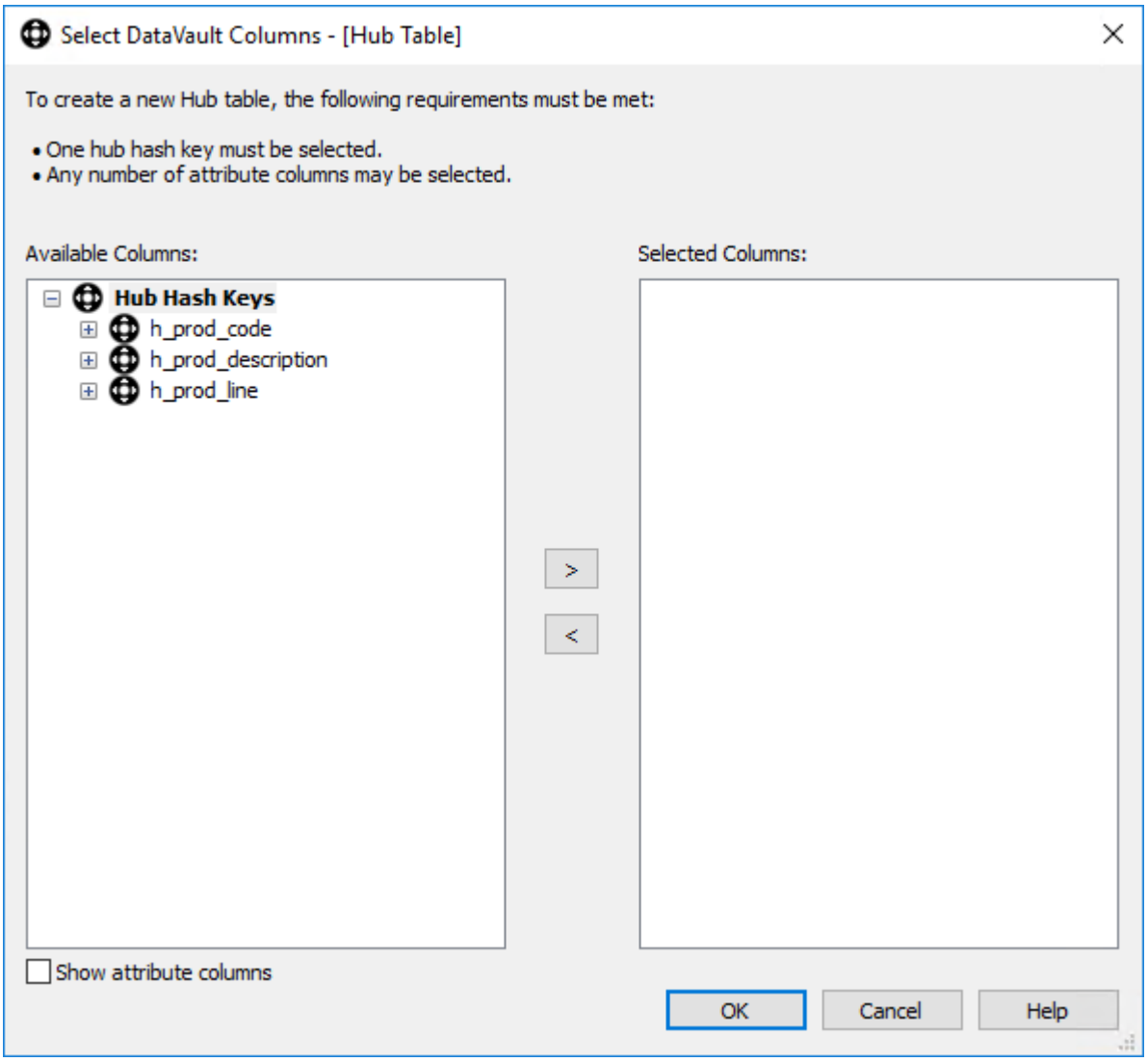
Notes

1. Custom aliases for the **dss_change_hash** column can be defined in the **Tools > Options > DSS Tables and Columns > DSS Columns to Include and Naming** window. If custom DSS aliases are defined then the aliases are used for the column names that are added to the new Load tables. Refer to **DSS Columns** for details.
2. The data type defined for the **dss_change_hash** column in **Tools > Options > Data Vault** window is applied when creating a satellite change hash key. Refer to **Data Vault Hash Keys setting** for details.
3. Refer to **Changing the Data Vault Hash Key Function in WhereScape RED 6.9.1.0** and above, if you need to amend the Hash key function on the shipped WhereScape RED **Data Vault Templates**.

© Copyright, WhereScape Software Ltd All rights reserved.

Hub, Link and Satellite Creation Wizard

This Wizard is launched when building a Data Vault object (**Hub**, **Link** or **Satellite** table) by dragging and dropping the source **Data Vault Stage** table from the right pane to the middle pane.

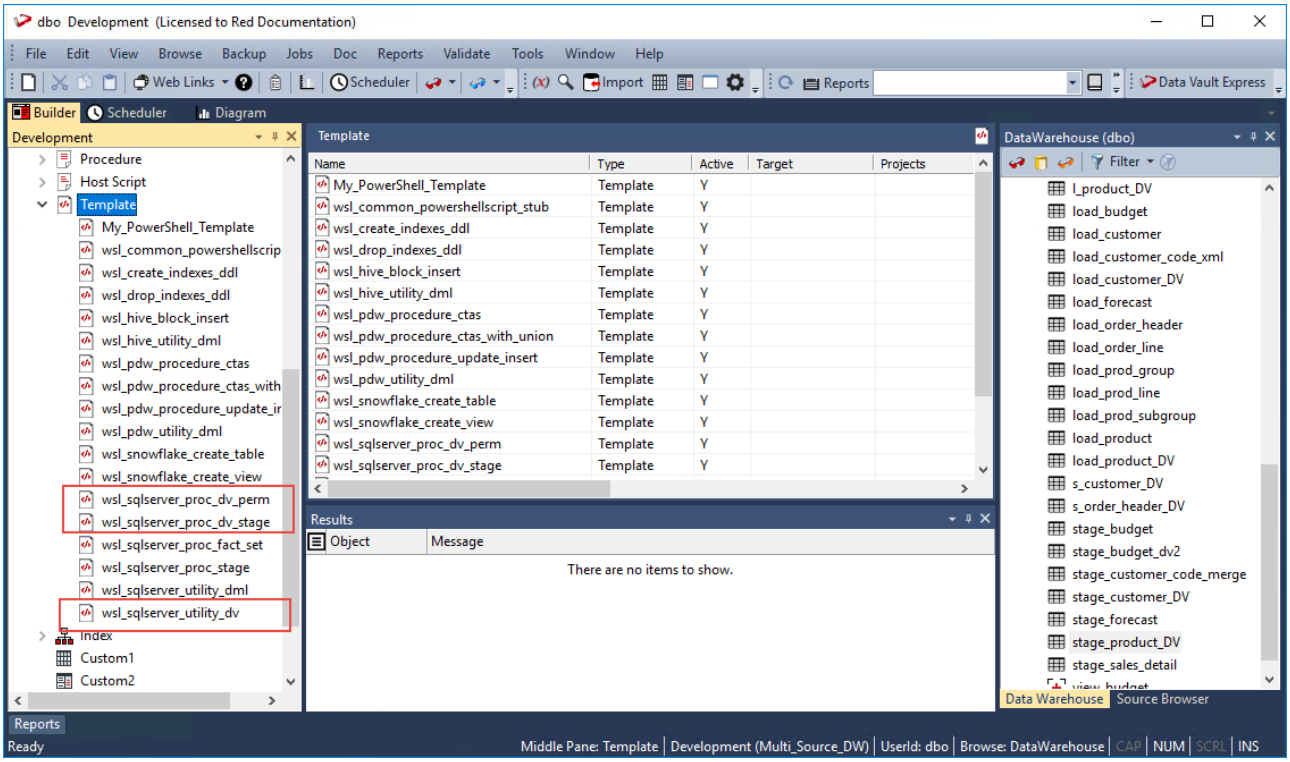


The detailed steps for using this Wizard is described **Creating the Hub, Link and Satellite Tables**.

© Copyright, WhereScape Software Ltd All rights reserved.

Data Vault Templates

Templates are used to generate update procedures for **Data Vault** objects. Users must select a template to use when generating the update procedure for **Data Vault** objects created in WhereScape RED.



The **SQL Server Data Vault** templates are described below:

- **wsl_sqlserver_proc_dv_stage** – this template creates an SQL Server procedure for updating WhereScape RED **Data Vault Stage** tables.
- **wsl_sqlserver_proc_dv_perm** – this template creates an SQL Server procedure for updating WhereScape RED **Data Vault** objects (**Hub**, **Link** and **Satellite** tables).
- **wsl_sqlserver_utility_dv** – this utility template contains generic SQL Server macros that are used by the other two templates above.

The above templates are available in WhereScape RED version 6.9.1.0 and above. If the templates are not visible in the **Template** objects list after installing/upgrading WhereScape RED, use the WhereScape RED Setup Administrator to Validate the MetaData Repository.

Notes:

- Templates for Oracle and Teradata are also supplied by WhereScape. For other database types, users need to create/provide their own templates.
- **Important:** The above templates have changed in WhereScape RED version 8.0.1.0. The Data Vault hash key functions for Teradata and Oracle repositories have been changed in the templates (wsl_teradata_utility_dv and wsl_oracle_utility_dv). Data Vault customers running on Teradata or Oracle who were using these templates from the RED 6.9.1.0 release are advised to retain that version of the template, unless they migrate any key generated using the previous hash key function.
  Refer to **Changing the Data Vault Hash Key Function in WhereScape RED 6.9.1.0 and above**, if you need to amend the Hash key function on the shipped WhereScape RED Data Vault Templates.

A Wizard to generate update procedures via templates is used to populate **Data Vault** tables. The detailed steps for using this Wizard is described in **Generating Update Procedures for Hub, Link and Satellite Tables**.

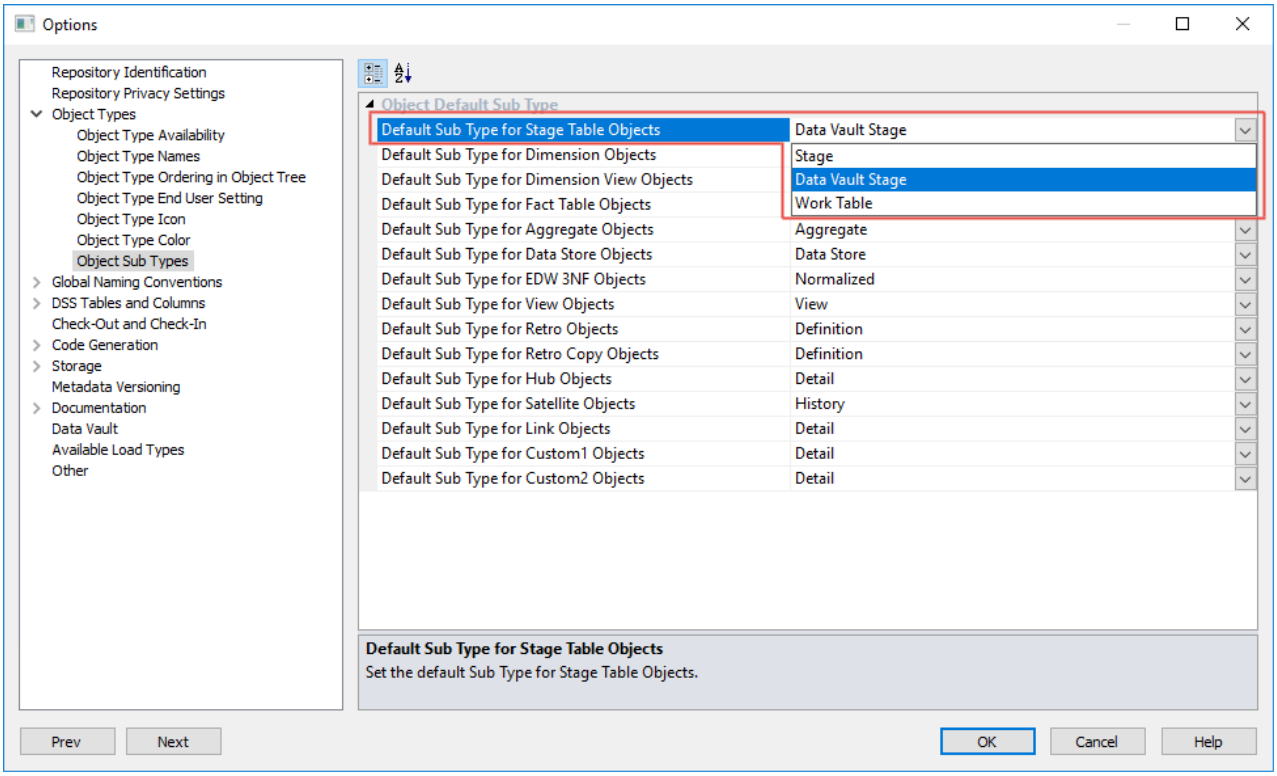
© Copyright, WhereScape Software Ltd All rights reserved.

Data Vault Settings

Settings for Data Vault objects can be configured from the **Tools > Options** screen.

Object Types Settings

The **Default Sub Type for Stage Table Objects** drop-down includes the **Data Vault Stage** option.



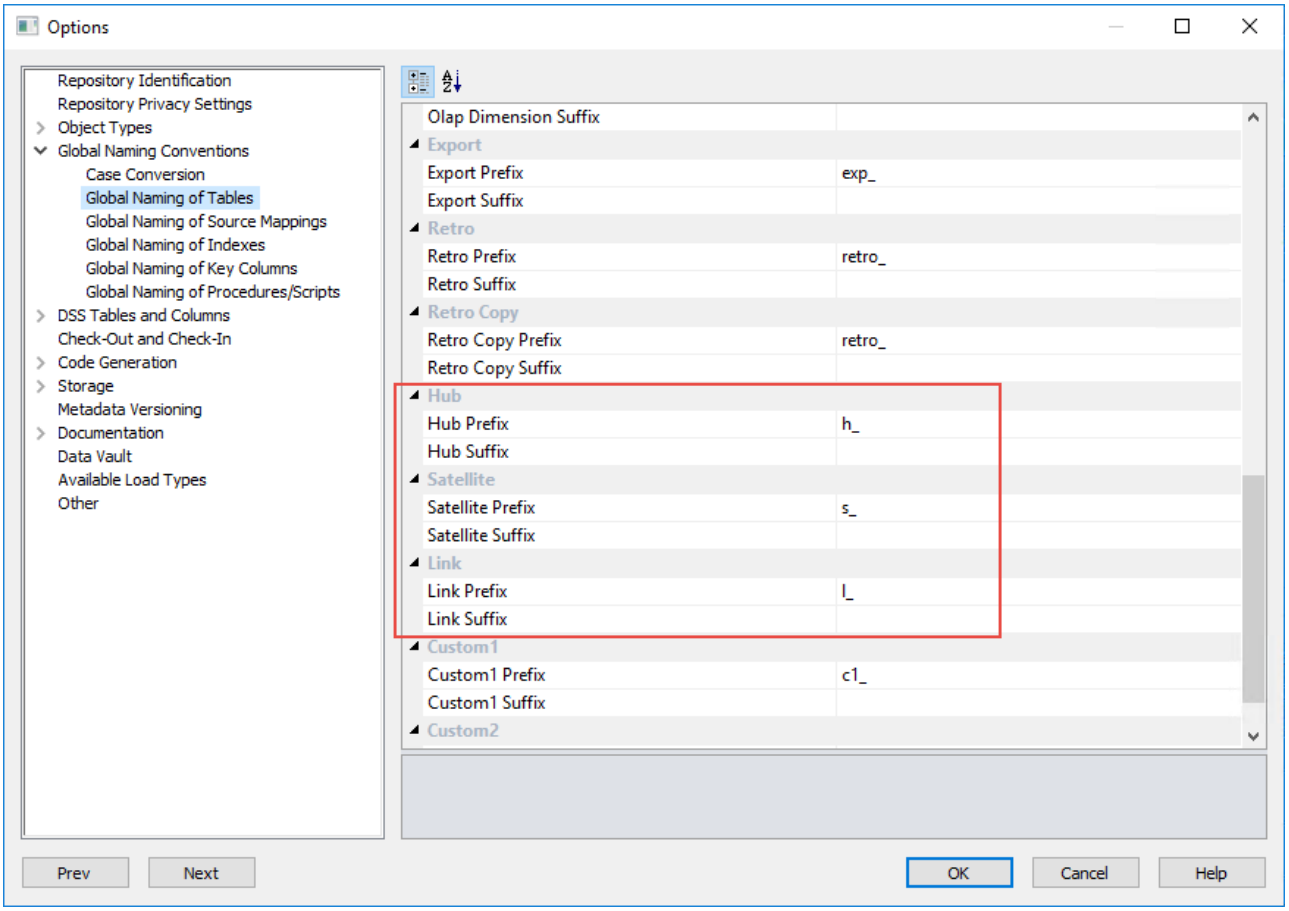
Configure this setting, if you want to set **Data Vault Stage** to be the default **Table Type** in the **Stage Table Properties** screen.

© Copyright, WhereScape Software Ltd All rights reserved.

Global Naming Conventions Settings

Global Naming of Tables

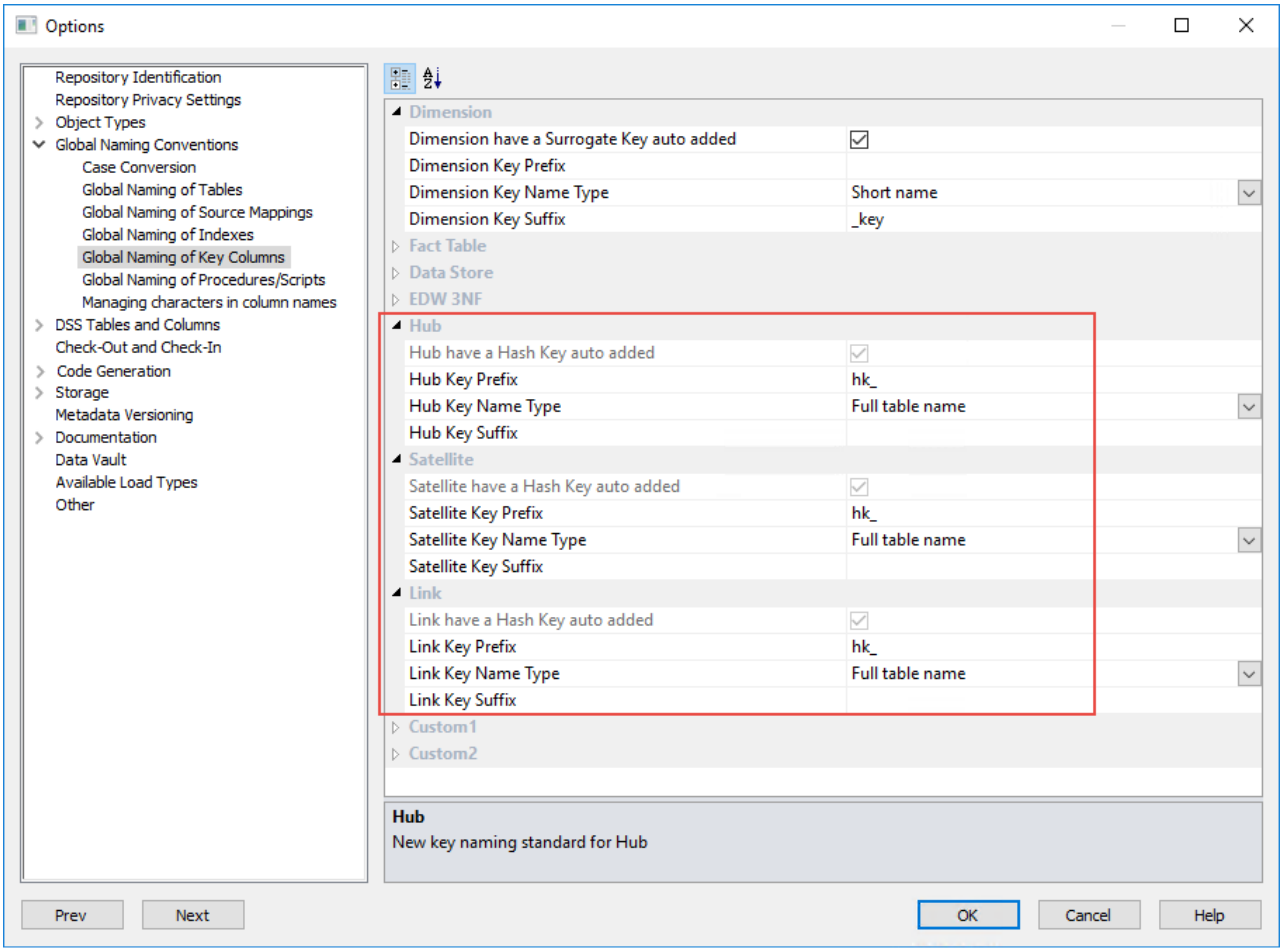
The **Global Naming of Tables** setting for **Hub**, **Satellite** and **Link** tables have been set to comply with the recommended standard naming convention for **Data Vault** tables. You can edit this setting to suit your requirements.



© Copyright, WhereScape Software Ltd All rights reserved.

Global Naming of Key Columns

The **Global Naming of Key Columns** setting for **Hub**, **Satellite** and **Link** tables enables you to define the default naming convention for the Hash Keys that are generated for these Data Vault objects (e.g. Key Prefix value plus the Source Column name). You can edit this setting to suit your requirements.



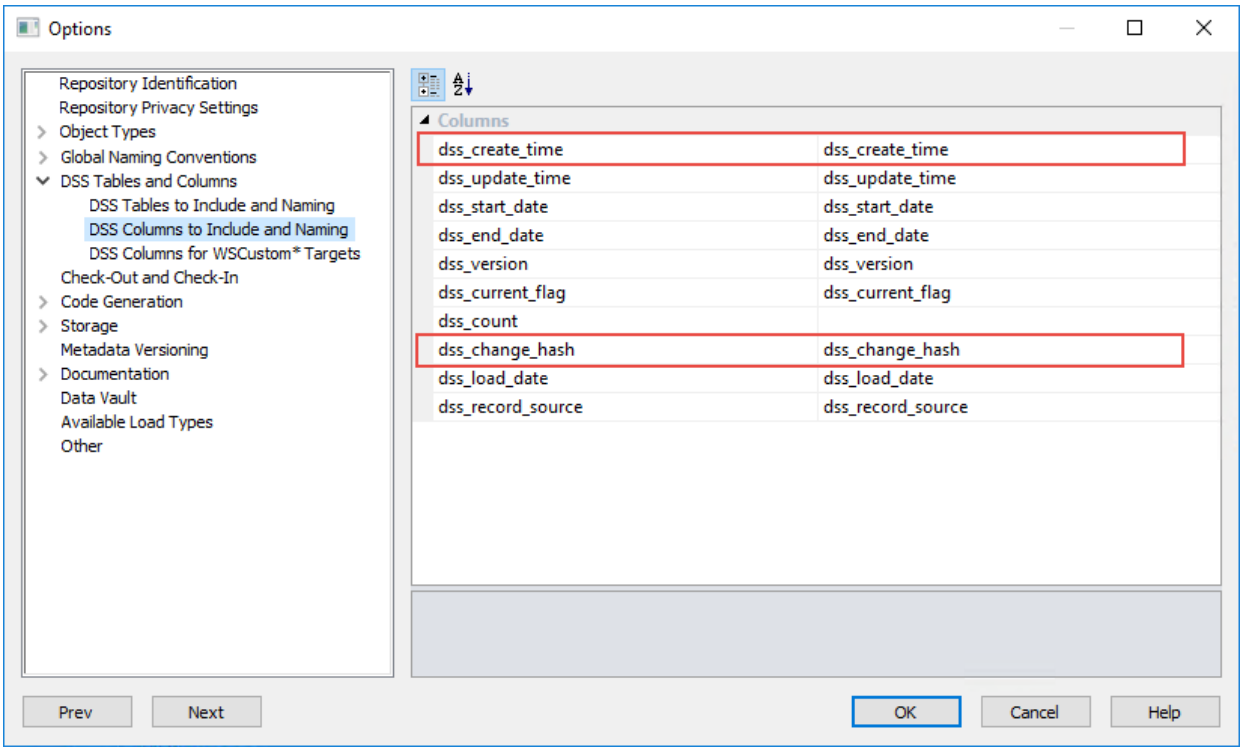
Note

WhereScape RED was enhanced to comply with Data Vault 2.0 standards. The Hash keys are now generated as part of the **Data Vault Stage table creation wizard** which are then passed during the drag and drop creation of new Data Vault objects. Hence hash key columns are now mandatory and must not be added during the creation of a Data Vault object.

© Copyright, WhereScape Software Ltd All rights reserved.

DSS Tables and Columns Settings

The **DSS Columns to Include and Naming** setting includes two additional columns which is described below:

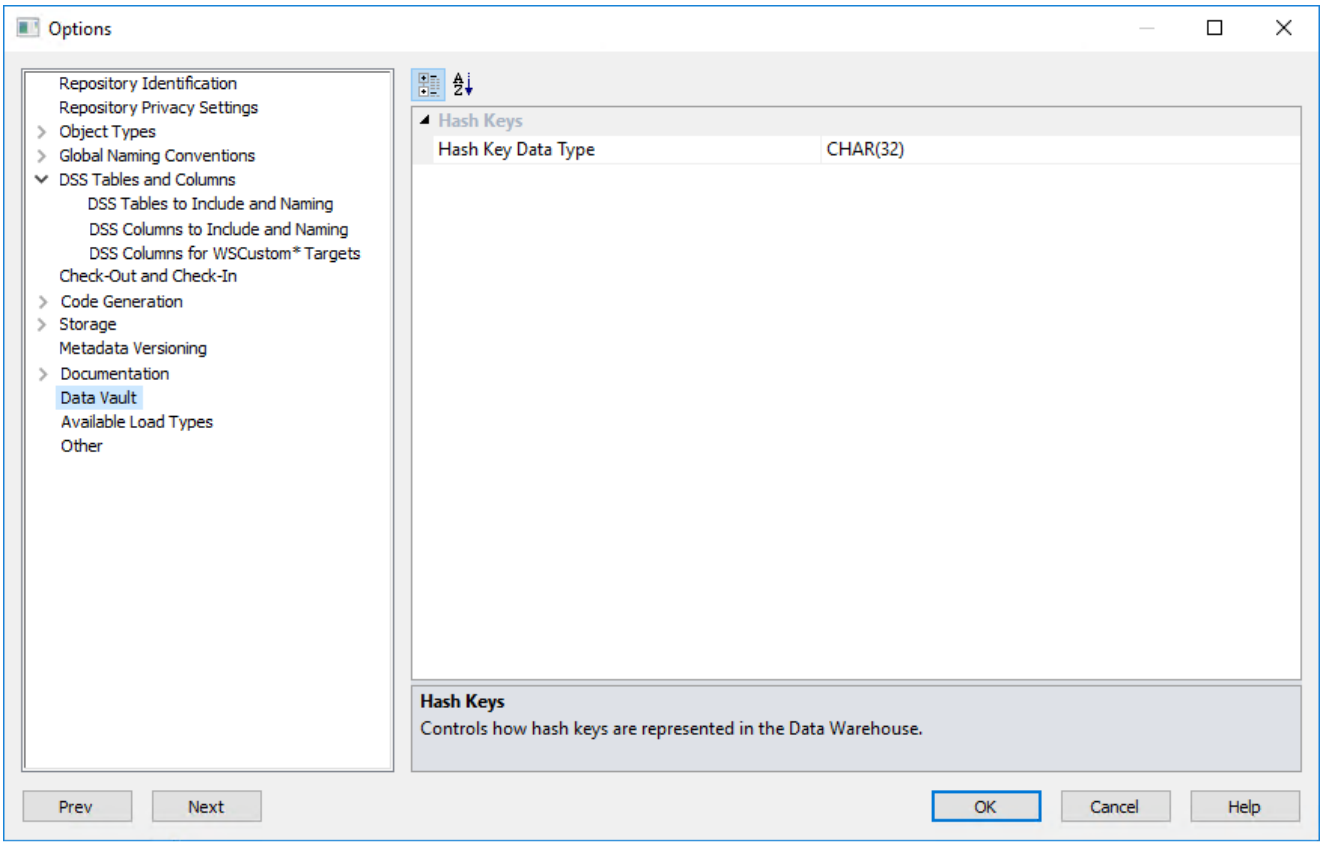


- **dss_create_time** – this column is added to all Stage, ODS, Normalized, Dimension, Fact and Aggregate tables for information purposes only. Leave blank to deactivate.
- **dss_change_hash** – this column is used to identify the differences in the descriptive columns of a Satellite table which is required for generating the change hash key for a Satellite table.

© Copyright, WhereScape Software Ltd All rights reserved.

Data Vault

The **Data Vault** setting is used to define the data type for the generated Hash Keys in RED that are used to build the Hub, Link and Satellite table objects.



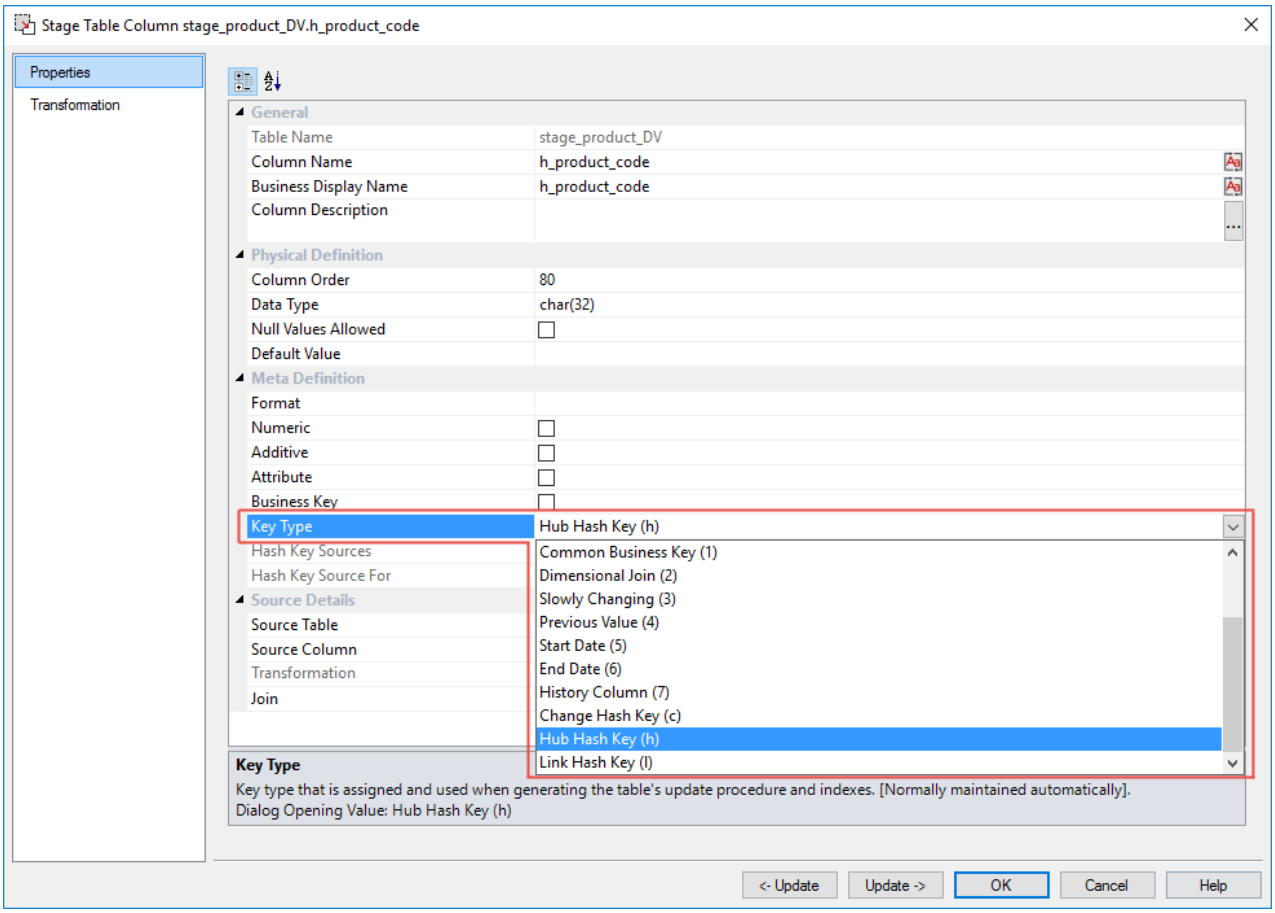
Note

Changing this data type has no impact on any existing hash key columns.

© Copyright, WhereScape Software Ltd All rights reserved.

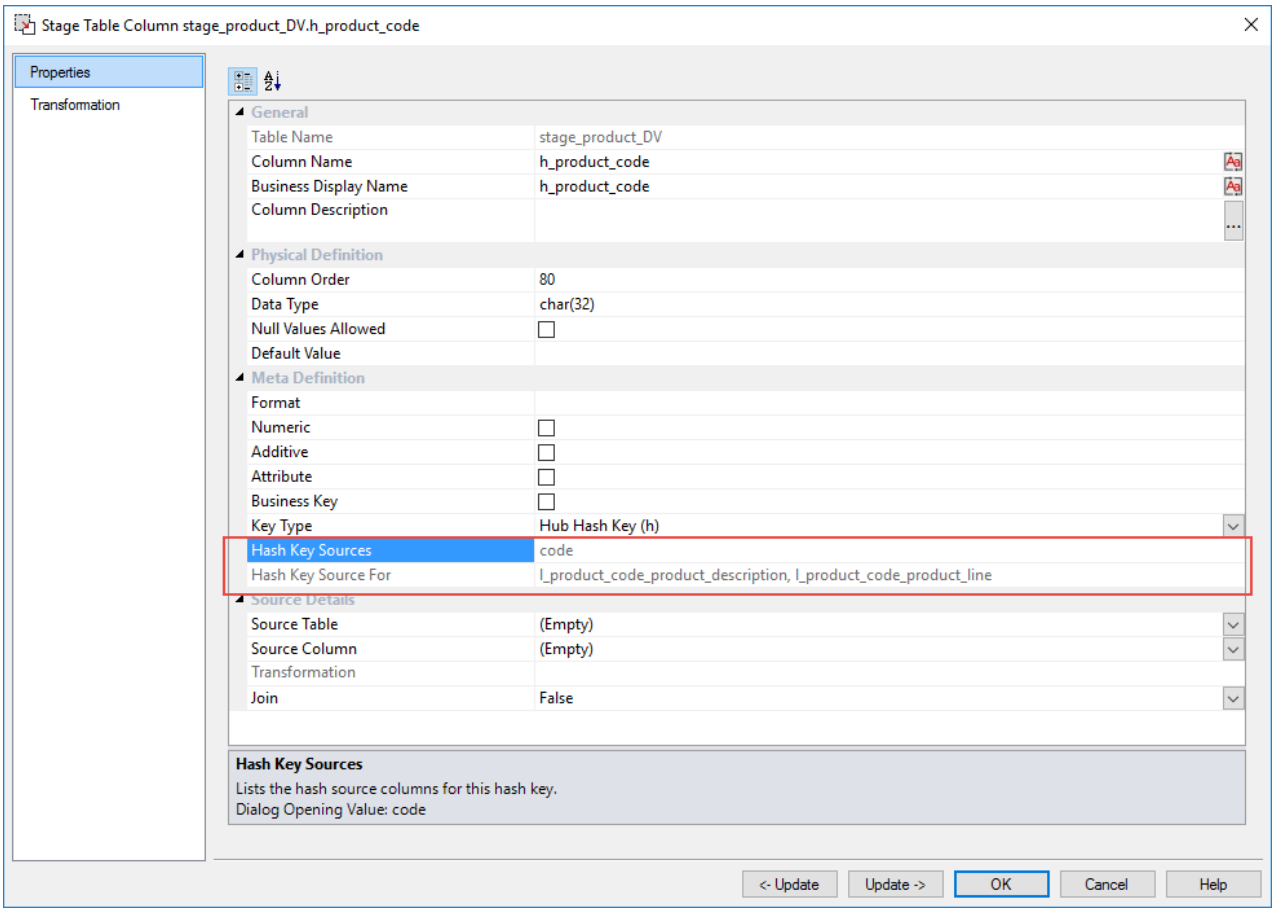
Table Column Properties

The **Key Type** field drop-down include options for **Data Vault** hash keys, e.g. **Change Hash Key (c)**, **Hub Hash Key (h)** and **Link Hash Key (l)**.



© Copyright, WhereScape Software Ltd All rights reserved.

Hash Key source information are also displayed for these key types.



- **Hash Key Sources** – displays the source columns that are used to generate the selected **Hub**, **Link** or **Change** hash key.
- **Hash Key Source For** – displays the hash keys columns that use the displayed hash key sources.

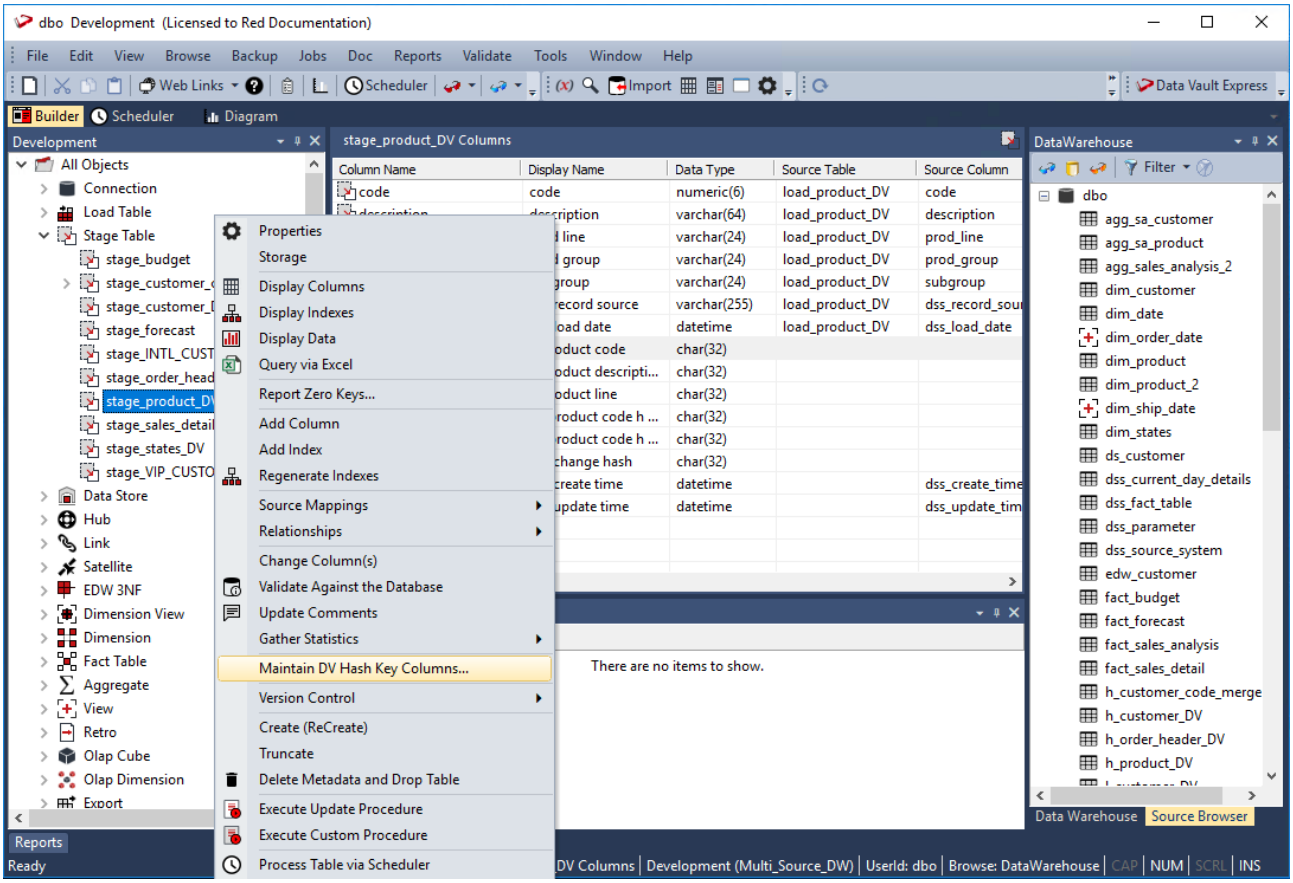
The hash key generation Wizard enables you to specify the source columns to be used in defining the **Hub**, **Link** and **Change** hash key columns. The detailed steps for using this Wizard is described in **Creating Data Vault Stage Tables**.

Once the hash key columns have been defined, another Wizard is used to generate the procedure to populate the columns. The detailed steps for using this Wizard is described in **Generating Update Procedures for the Data Vault Stage Table**.

© Copyright, WhereScape Software Ltd All rights reserved.

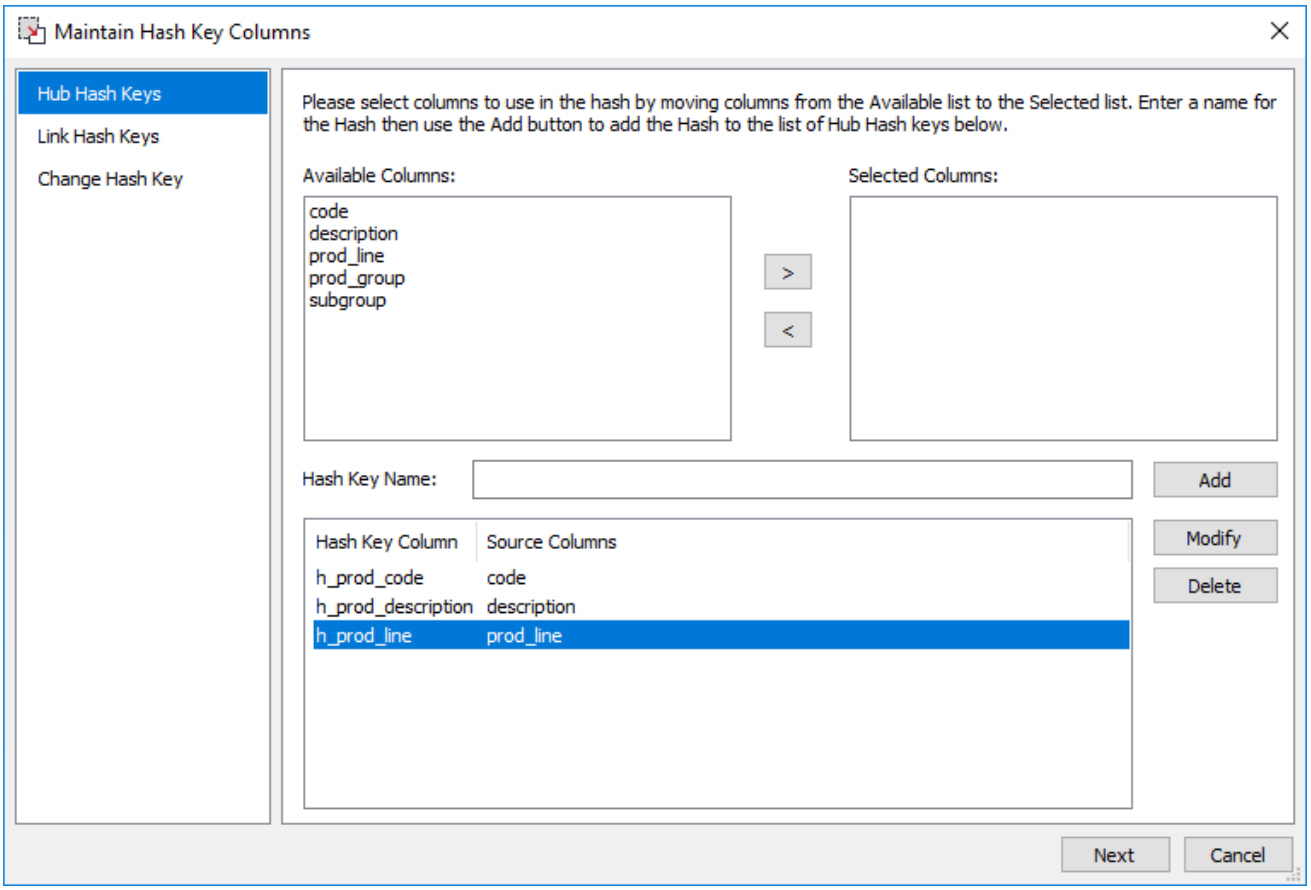
Maintain Hash Key Columns

The context menu for **Stage Table** objects, listed in the left pane provides an option for maintaining **Data Vault** hash key columns.



You can review the composition of existing hash keys for a **Data Vault Stage** table (Hub, Link and Satellite) and create additional hash keys by selecting the **Maintain DV Hash Key Columns** option from the selected **Data Vault Stage Table**'s context menu. This launches the hash key generation Wizard which enables you to maintain the source columns defined for the hash keys. Refer to **Creating Data Vault Stage Tables** for details.

© Copyright, WhereScape Software Ltd All rights reserved.



Note

To remove or change a hash key column, you need to delete it first, e.g. right click the column listed in the middle pane and then select **Delete Column** from the context menu.

Building Data Vault Objects

To build Data Vault objects (Hub, Link and Satellite) in WhereScape RED, involves the following procedures.

1. Creating Load Tables with the required DSS columns.
2. Creating Data Vault Stage tables.
3. Generating update procedures for the Stage table via templates.
4. Creating the Hub, Link and Satellite tables.
5. Generating update procedures for the Hub, Link and Satellite tables via templates.

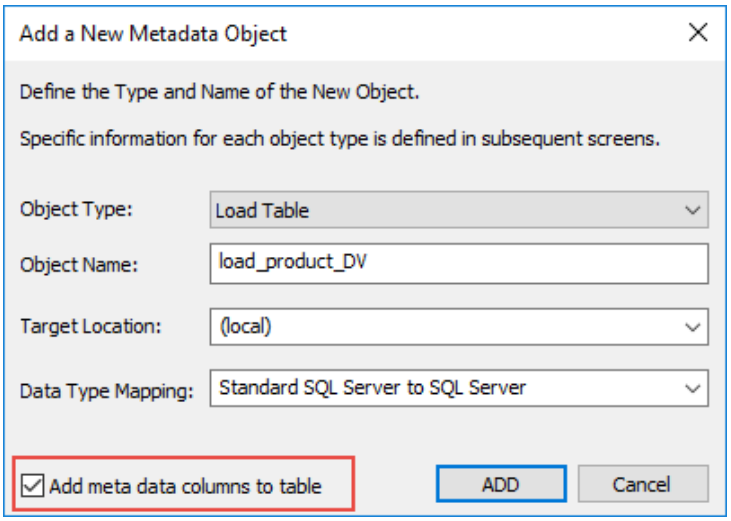
The detailed steps for each procedure are outlined in the following sections.

Creating Load Tables

The following describe the steps for creating a **Load** table:

1. Browse to the source system connection required (**Browse > Source Tables**).
2. Double-click the **Load Table** object group in the left pane, the middle pane displays a list of existing Load tables.
3. Click the source table from the right pane and drag it to the middle pane. You need to create the **Load** table with the required DSS columns—the option to add default meta data columns to the load table must be selected:

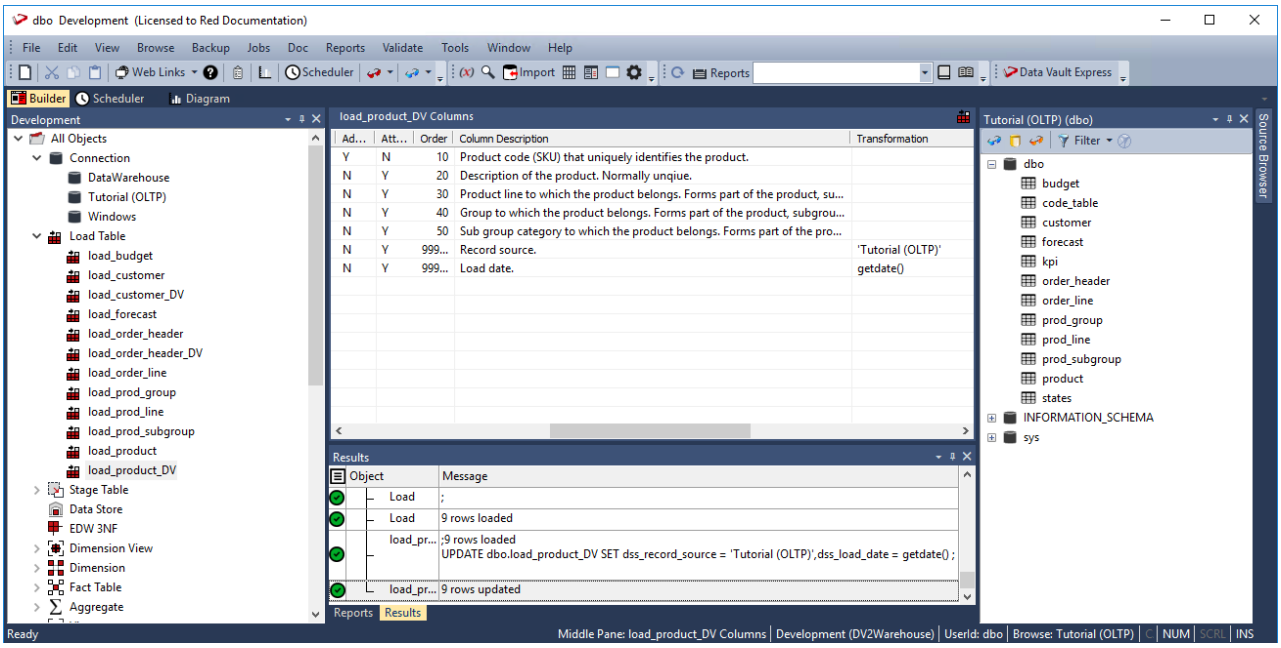
© Copyright, WhereScape Software Ltd All rights reserved.



When the build table operation is performed, the Load table created has the two additional columns which are populated by transformations:

- **dss_record_source** – the connection or source for the load table.
- **dss_load_date** – the date when the data was loaded to the table. This is updated every time a load operation is performed.

These DSS columns added, include column description and transformation information.



Notes

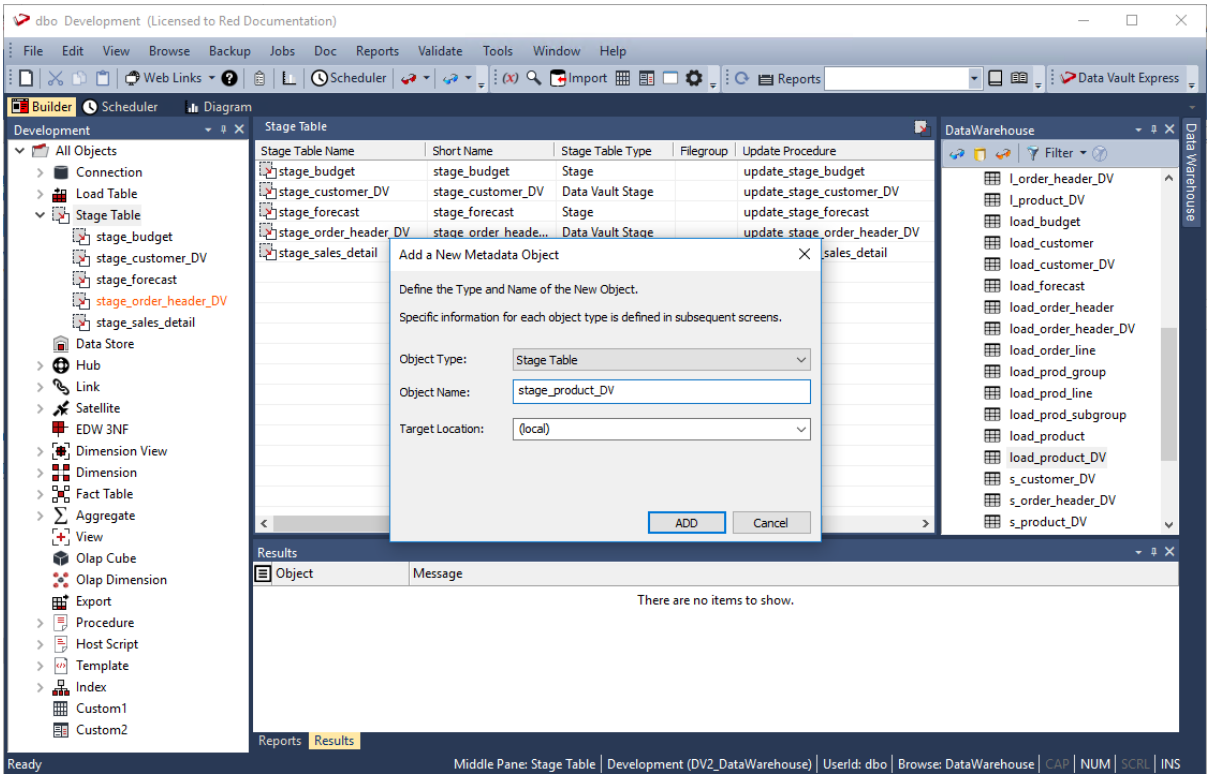
1. Custom aliases for the **dss_load_date** and **dss_record_source** columns can be defined in the **Tools > Options > DSS Tables and Columns > DSS Columns to Include and Naming** window. If custom DSS aliases are defined, then the aliases are used for the column names that are added to the new Load tables. Refer to **DSS Columns** for details.
2. The **dss_record_source** column transformation is set to the name of the source connection when the table is first created. If the source connection changes, the transformation value does not update automatically. You can manually update the transformation—refer to **Transformations** for details

© Copyright, WhereScape Software Ltd All rights reserved.

Creating Data Vault Stage Tables

The following describe the steps for creating a **Data Vault Stage** table:

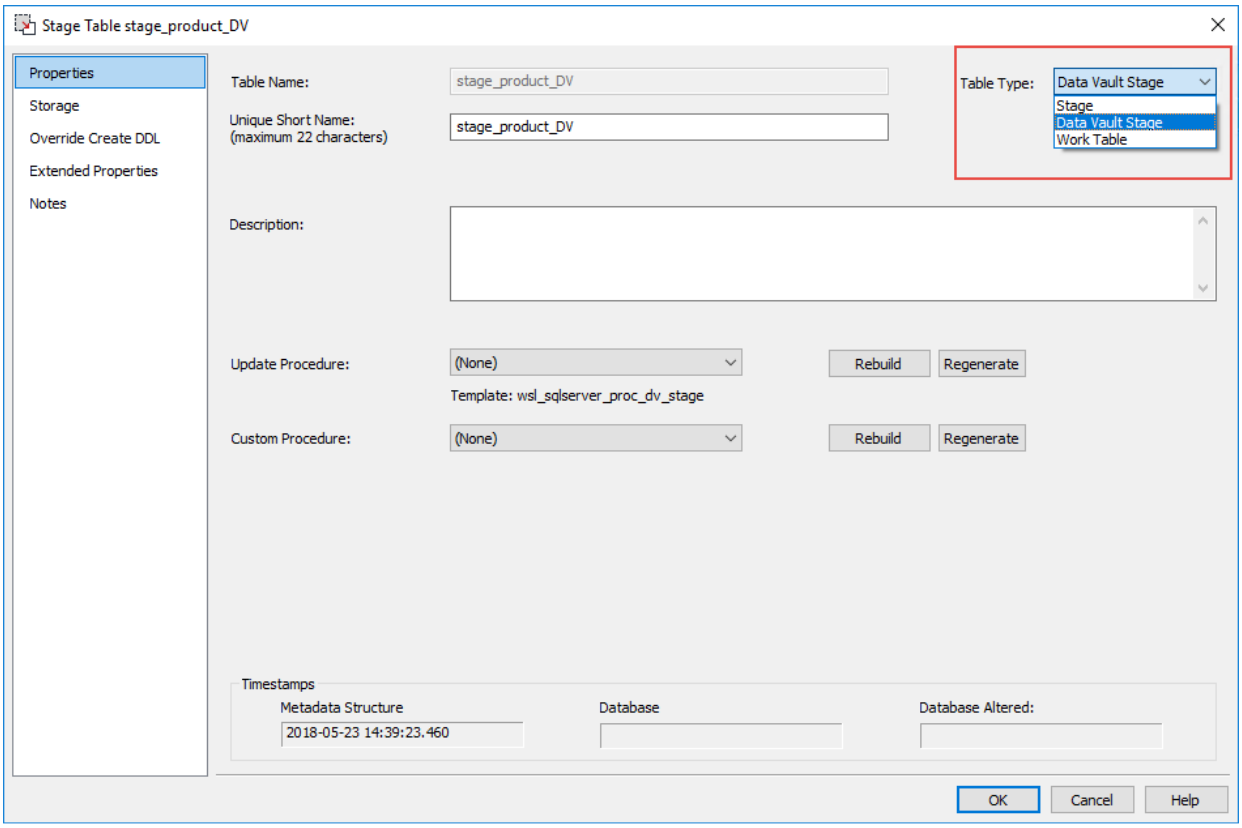
1. Browse to the Data Warehouse (click **Browse > Data Warehouse**) connection to create the **Data Vault Stage** table.
2. Double-click the **Stage** object group in the left pane, the middle pane displays a list of existing **Stage** tables.
3. Click the source **Load** table from the right pane and drag it to the middle pane. The selected **Load** table must have the required DSS columns (**dss_record_source** and **dss_load_date**).



4. The **Add a New Metadata Object** screen appears and classifies the new object as a **Stage** table. It provides a default name based on the **Load** table name. Accept the name or enter a new name for the **Stage** table and click **ADD** to continue.

© Copyright, WhereScape Software Ltd All rights reserved.

5. On the **Table Properties** screen, select the **Data Vault Stage** option from the **Table Type** drop-down.

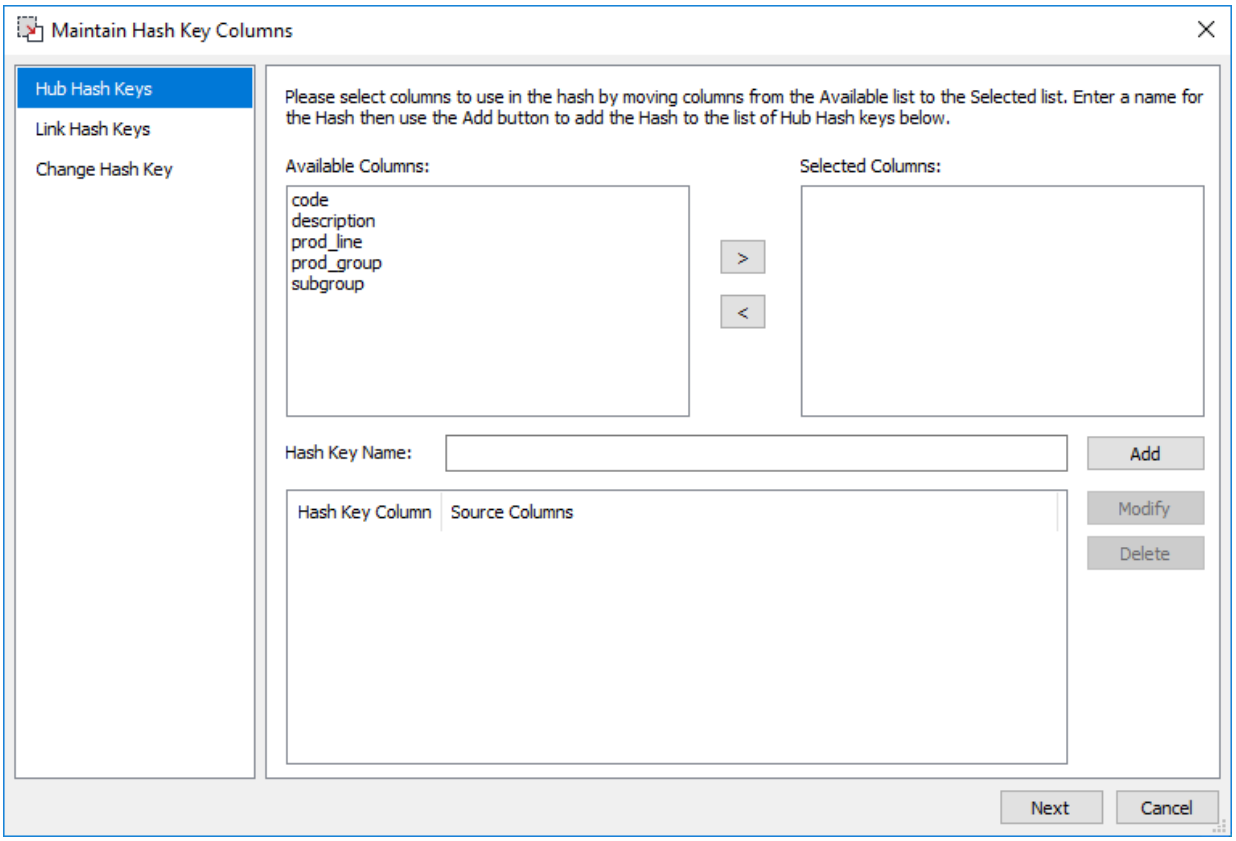


Notes

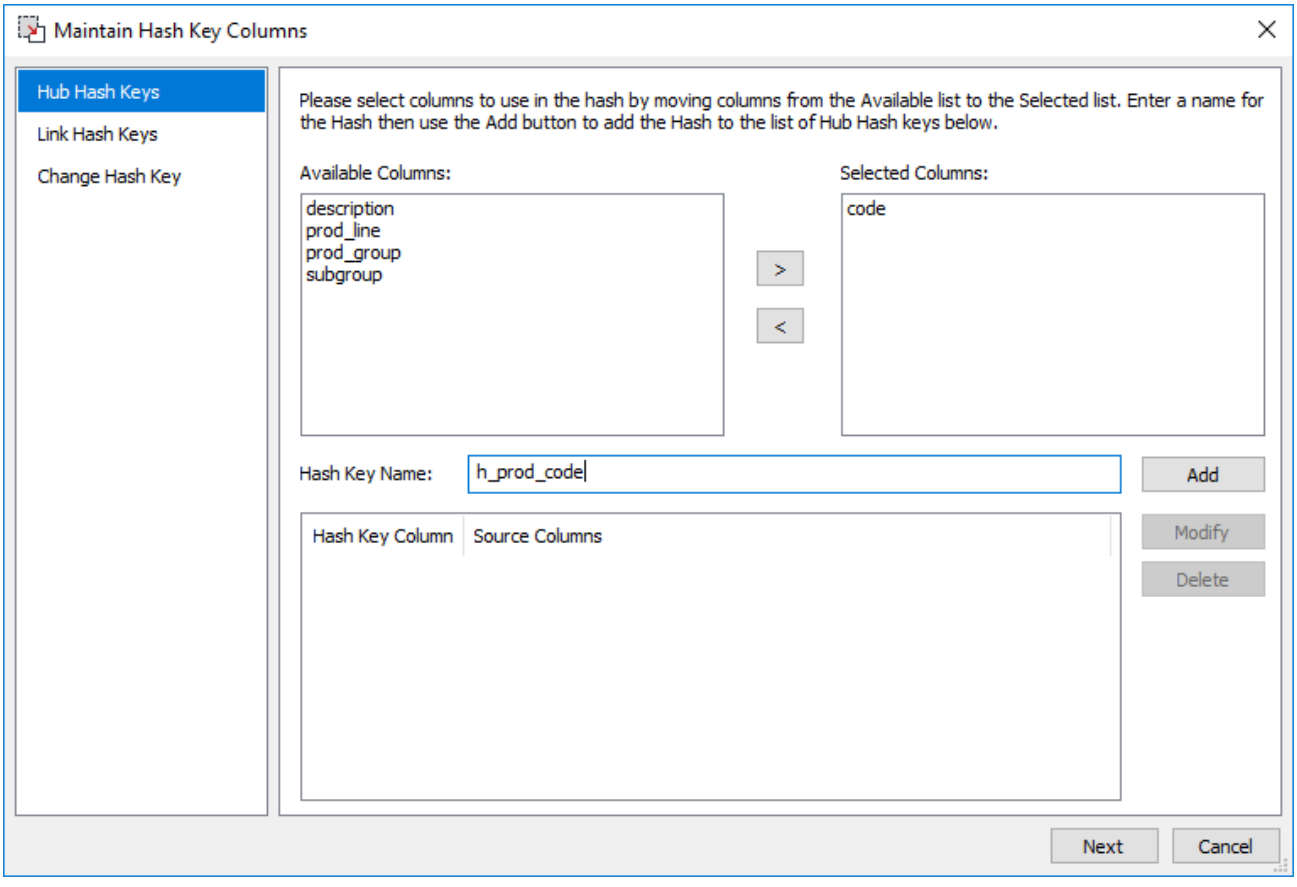
1. You can set this table type to be the default by configuring the **Default Sub Type for Stage Table Objects** setting in the **Tools > Options > Object Types > Object Sub Types** screen.
2. RED displays the name of the previously used update procedure template by default, below the **Update Procedure** drop-down.

6. Click **OK** in the table **Properties** screen to launch the Wizard that enables you to define the source columns for the **Hub**, **Link** and **Change** hash key columns.

© Copyright, WhereScape Software Ltd All rights reserved.



7. Select the source column(s) to use in defining the first Hub hash key. The **Hash Key Name** is formed based on the prefix (defined in the **Tools > Options > Global Naming Conventions > Global Naming of Key Columns** settings) and the source column(s) name. You can manually change the name if required.



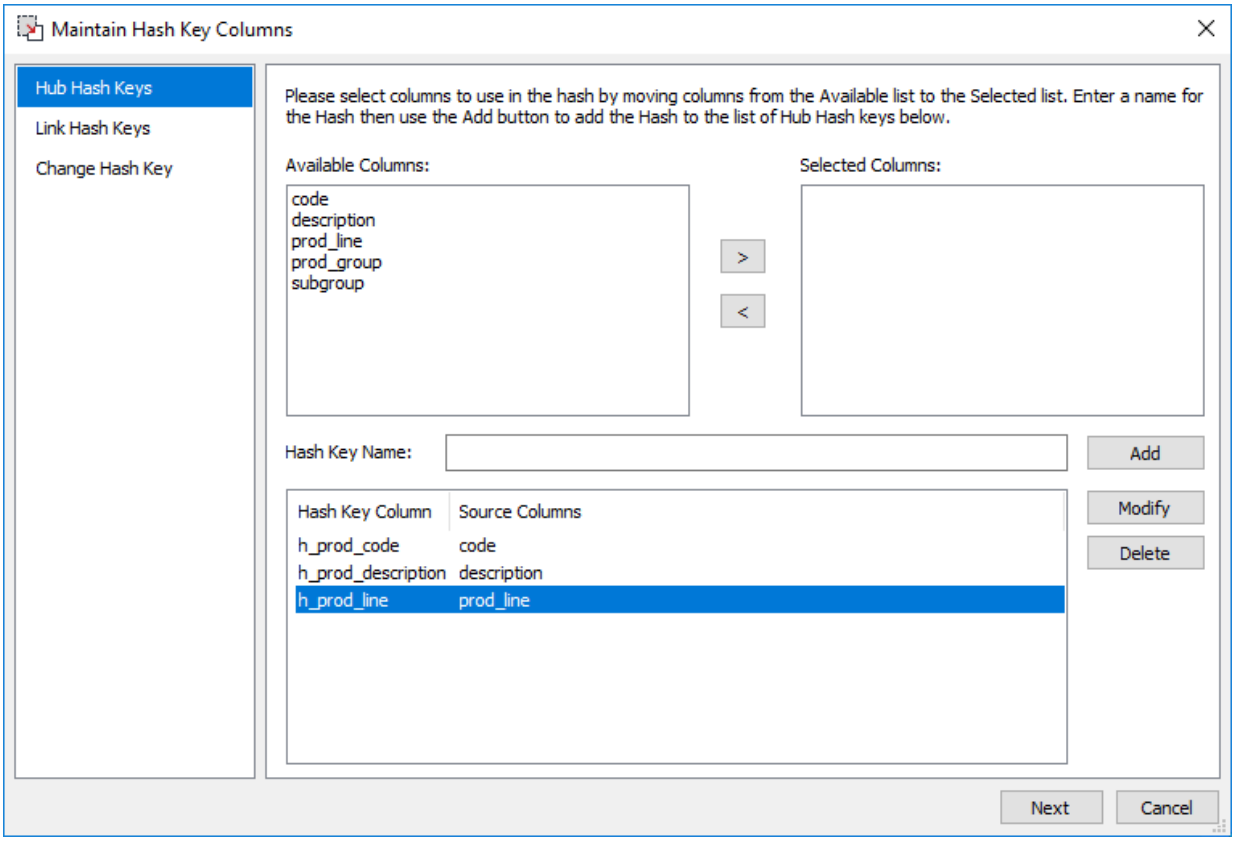
© Copyright, WhereScape Software Ltd All rights reserved.

8. Click **Add** to create the first Hub hash key. Repeat the same steps as required. Use the **Modify** button to edit a selected hash key. The **Delete** button enables you to delete a selected hash key. Repeat the same steps as required.

Note

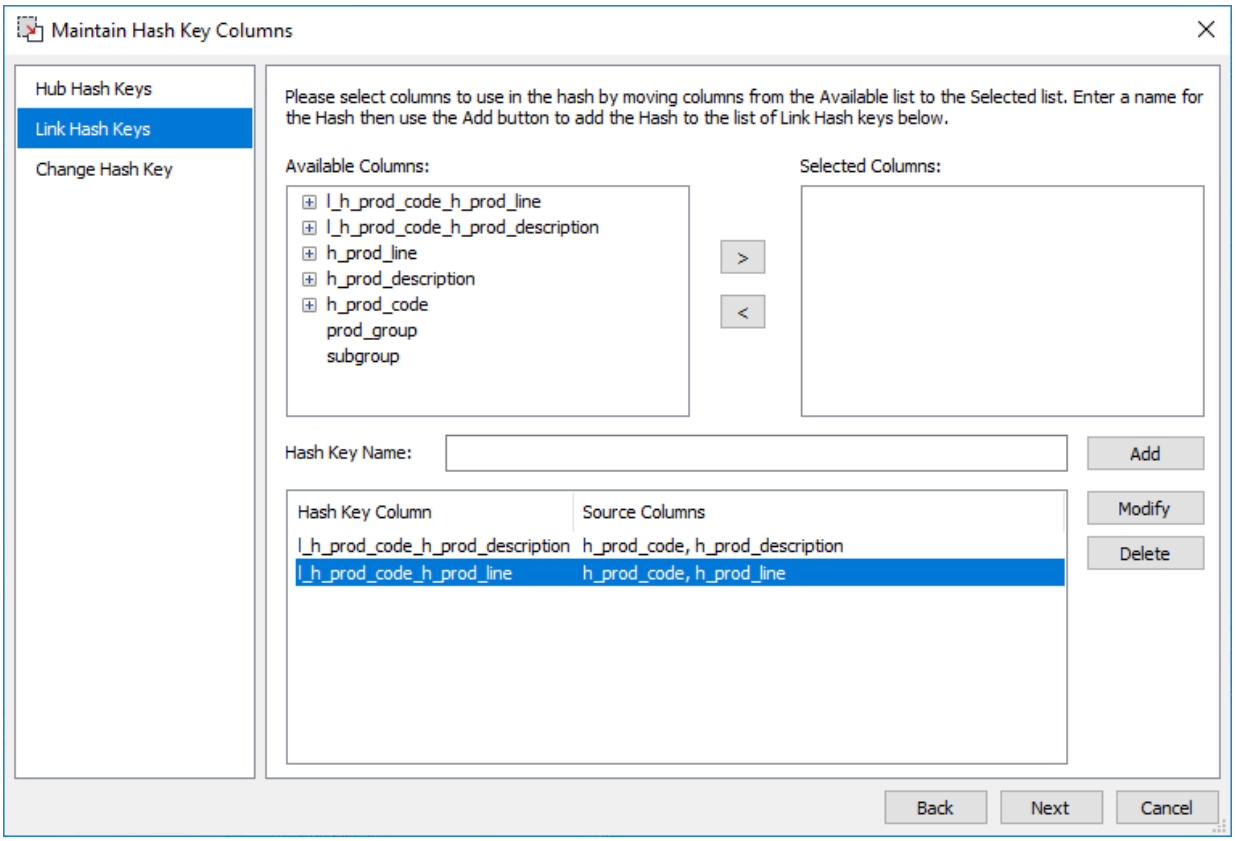
A confirmation message is displayed when you attempt to modify or delete a hash key that is being used by other objects in RED.

9. Once all the required **Hub Hash Keys** are created, click **Next** to progress to the **Link** hash key generation screen.

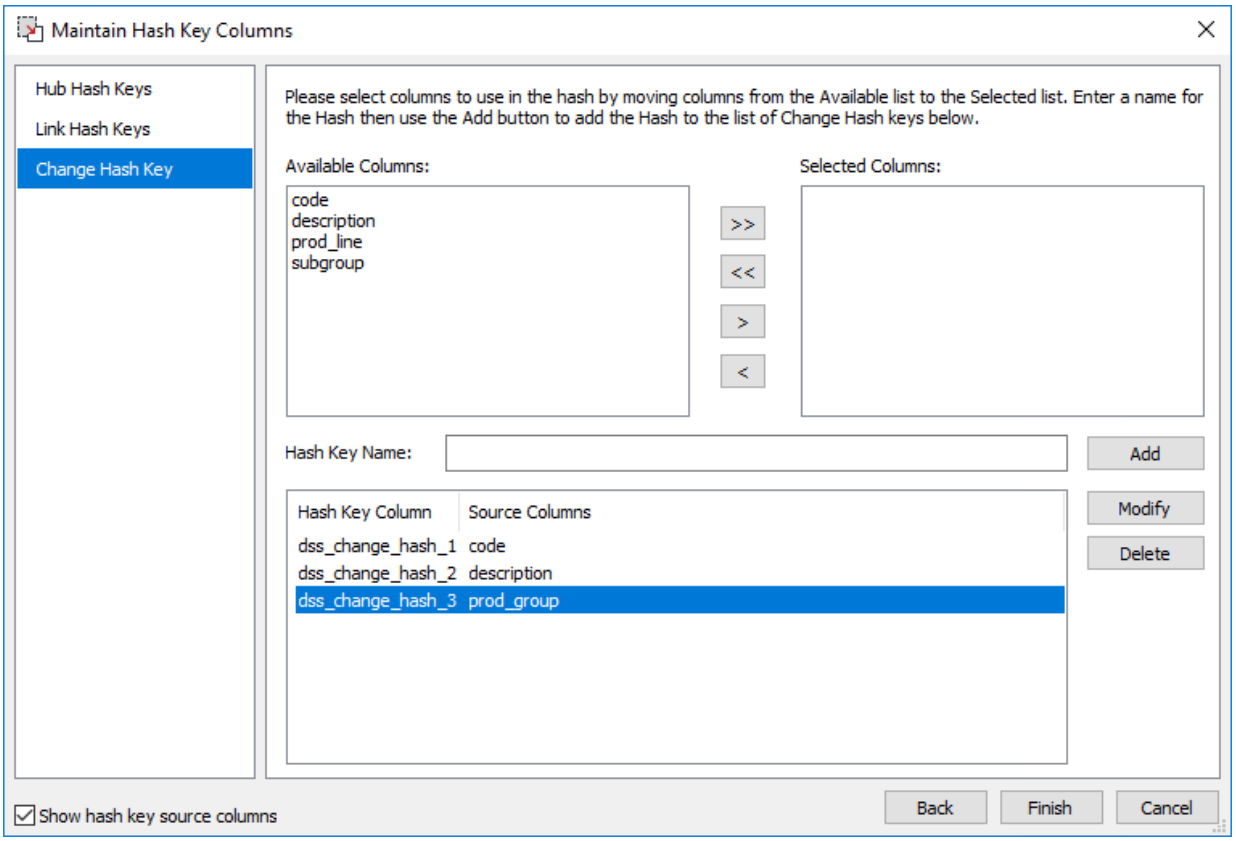


10. Creating the **Link Hash Keys** involves the same process, follow the previous steps (6 and 7) to select *multiple* source columns to combine to create the **Link** hash key. Click **Add** to create the first Link hash key. Repeat the same steps for any subsequent keys. Once all the required **Link Hash Keys** are created, click **Next** to progress to the **Change** hash key generation screen.

© Copyright, WhereScape Software Ltd All rights reserved.



11. Follow the same steps to select the columns to use for the **Change Hash Key**. You can select the **Show hash key source columns** check box below the screen to display and use hash key source columns in defining the Change Hash keys.

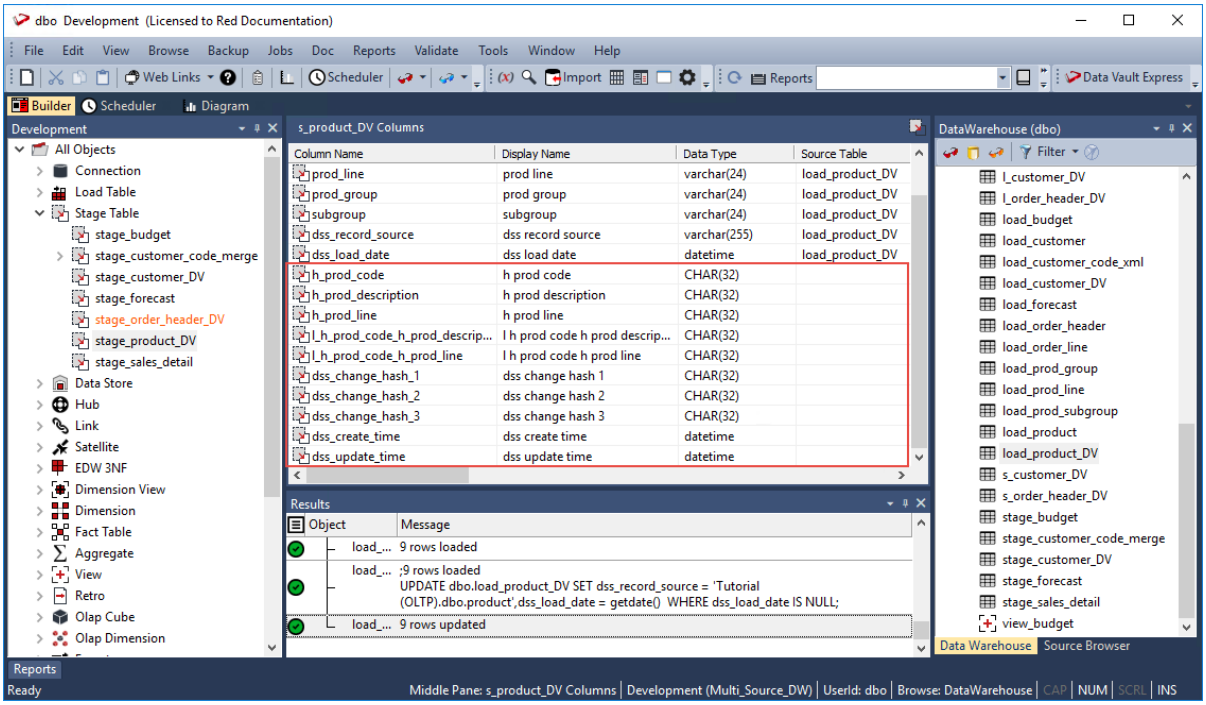


© Copyright, WhereScape Software Ltd All rights reserved.

Notes

1. Custom aliases for the **dss_change_hash** column can be defined in the **Tools > Options > DSS Tables and Columns > DSS Columns to Include and Naming** window. If custom DSS aliases are defined then the aliases are used for the column names that are added to the new Load tables. Refer to **DSS Columns** for details.
2. The data type defined for the **dss_change_hash** column in **Tools > Options > Data Vault** window is applied when creating a satellite change hash key. Refer to **Data Vault Hash Keys setting** for details.

12. Click **Finish**, once you have defined the required list of descriptive columns for the Change hash key. The new **Data Vault Stage** table is added to the **Stage Table** objects list in the left pane and the columns included in the table are listed in the middle pane.



In addition to the columns defined from the Load table, the following columns and their metadata have been added to the Data Vault Stage table:

- The Hub Hash Keys
- The Link Hash Keys
- The Change Hash Key
- The DSS_CREATE_TIME column
- The DSS_UPDATE_TIME column

The metadata for the hash columns include the source columns that were used to create them (used to generate the hash keys).

The hash keys created are used in the subsequent Data Vault object (Hub, Links and Satellites) creation Wizards.

Tip

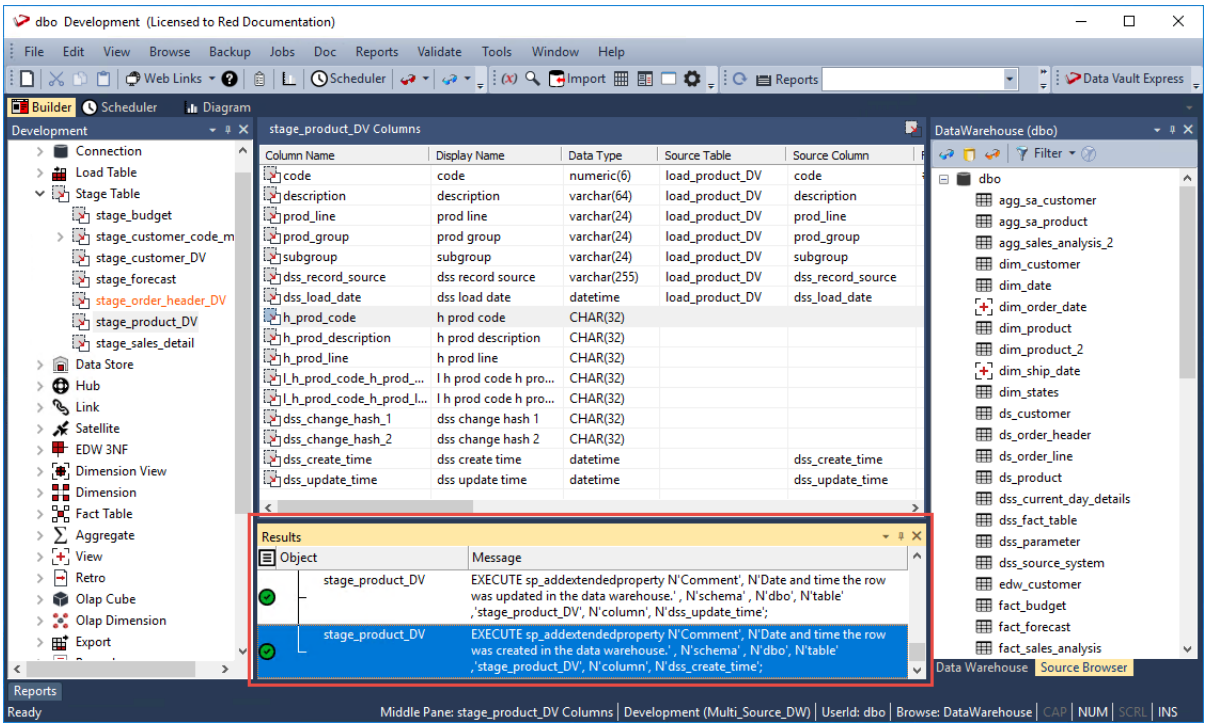
You can review the composition of existing hash keys for a Data Vault Stage table (Hub, Link and Satellite) and create additional hash keys by selecting the **Maintain DV Hash Key Columns** option from the selected **Data Vault Stage** table's context menu. This launches the hash key generation Wizard which enables you to maintain the source columns defined for the hash keys.

Note

To remove or change a hash key column, you need to delete it first, e.g. right click the column listed in the middle pane and then select **Delete Column** from the context menu.

© Copyright, WhereScape Software Ltd All rights reserved.

13. Right-click the new **Data Vault Stage** table you defined from the left pane, under the **Stage Table** objects list and select **Create (ReCreate)** from the context menu to create the table. The **Results** pane displays confirmation that the **Data Vault Stage** table was successfully created.



Note

For Teradata users, click **Yes** on the Primary Index prompt and then specify **code** on the **Primary Index Columns** field of the table **Properties** screen.

Once the new Data Vault Stage table is defined and created, clicking the **Rebuild** button in the **Table Properties** screen launches the Wizard to generate the procedure to populate the table. This Wizard utilizes a template to create the procedure.

The detailed steps for using this Wizard is described in the next section, **Generating Update Procedures for the Data Vault Stage Table**.

© Copyright, WhereScape Software Ltd All rights reserved.

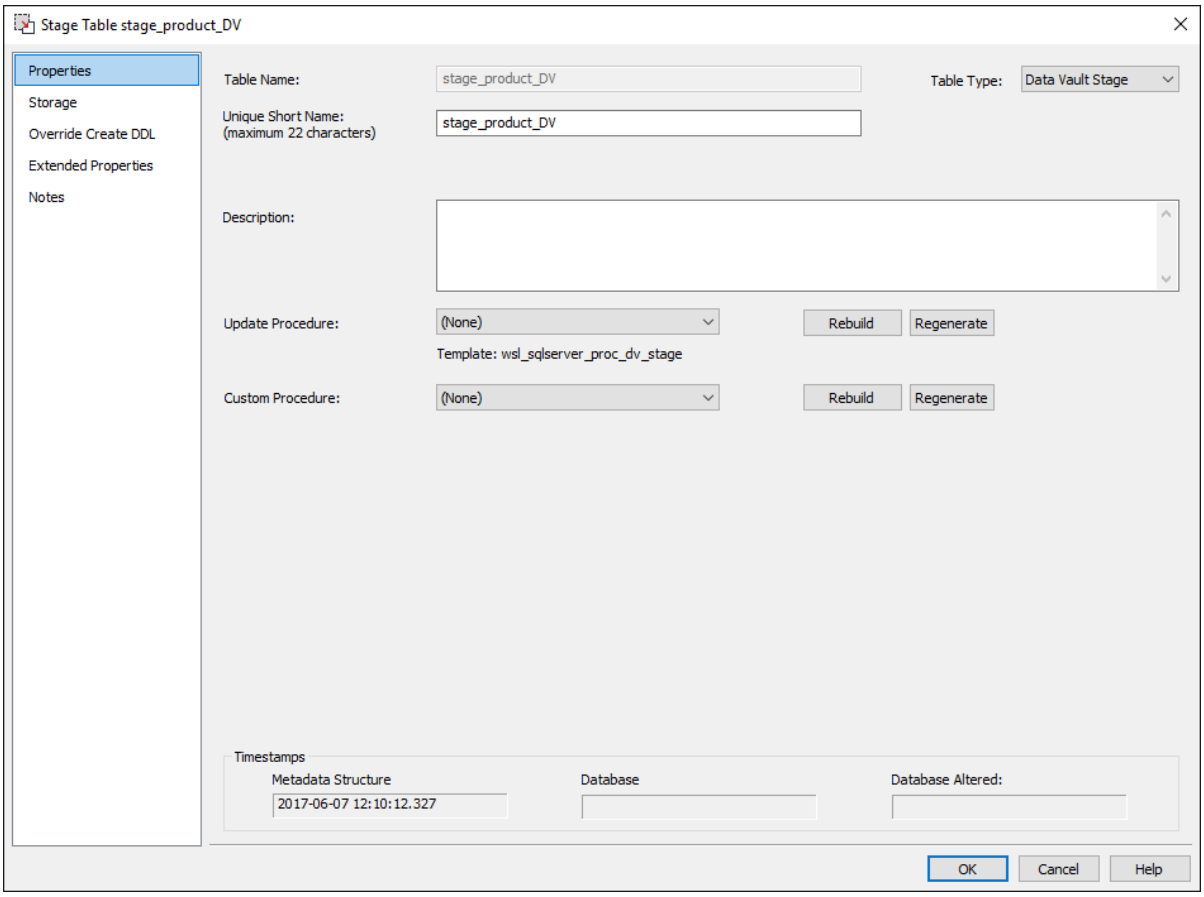
Generating Update Procedures for the Data Vault Stage Table

After successfully defining and creating the **Data Vault Stage** table, you can generate the update procedure via a template to populate the table.

Notes

- Please ensure that you have installed the WhereScape supplied templates (refer to **Data Vault Templates**) or created your own Data Vault templates, before performing the steps below.
- **Important:** The WhereScape supplied templates have changed in WhereScape RED version 8.0.1.0. The Data Vault hash key functions for Teradata and Oracle repositories have been changed in the templates (wsl_teradata_utility_dv and wsl_oracle_utility_dv). Data Vault customers running on Teradata or Oracle who were using these templates from the RED 6.9.1.0 release are advised to retain that version of the template, unless they migrate any key generated using the previous hash key function.

1. Click the **Rebuild** button in the table **Properties** screen to launch the procedure generation Wizard to populate the table.

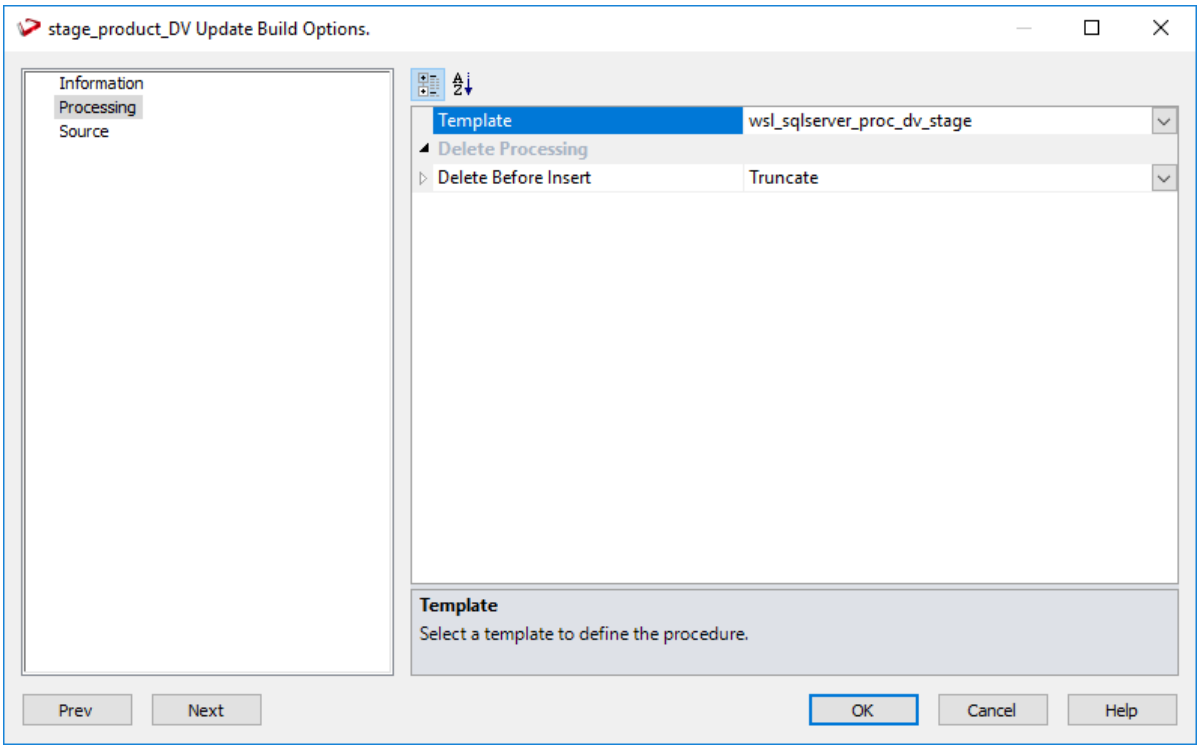


Note

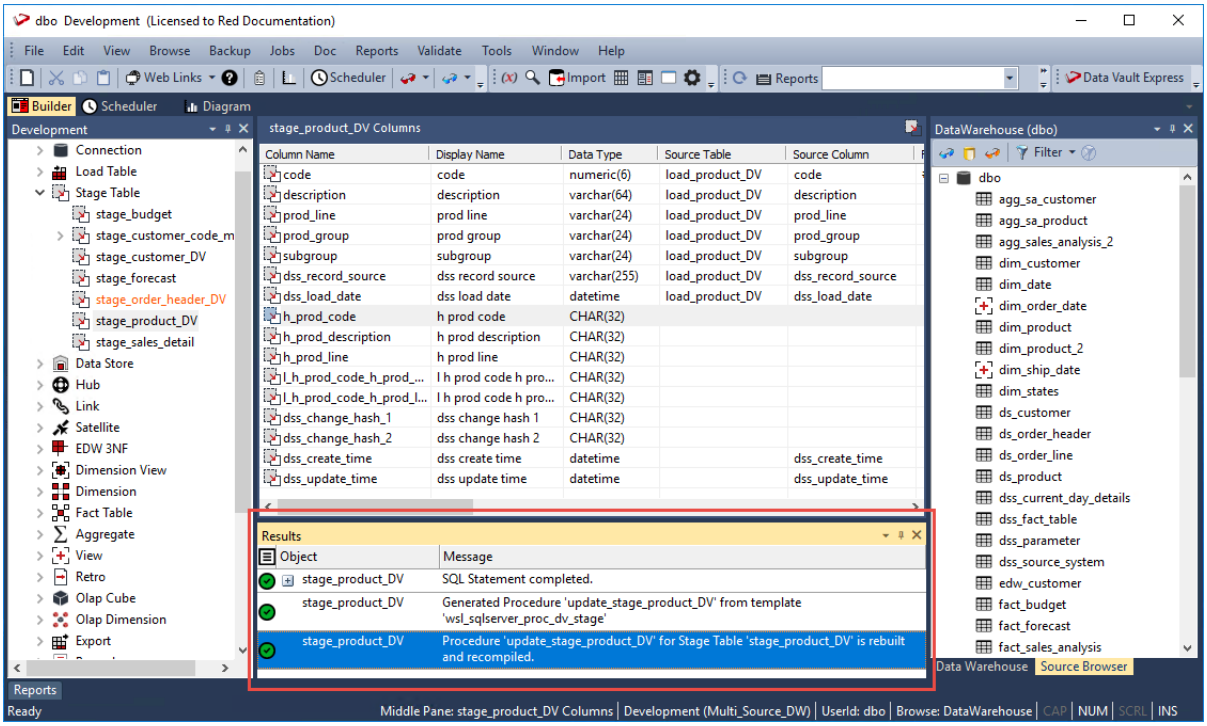
RED displays the name of the previously used update procedure template by default, below the **Update Procedure** drop-down.

2. On the **Processing** tab of **Table Update Build Options** screen, select the template to use from the **Template** drop-down or use the previous update procedure template.

© Copyright, WhereScape Software Ltd All rights reserved.

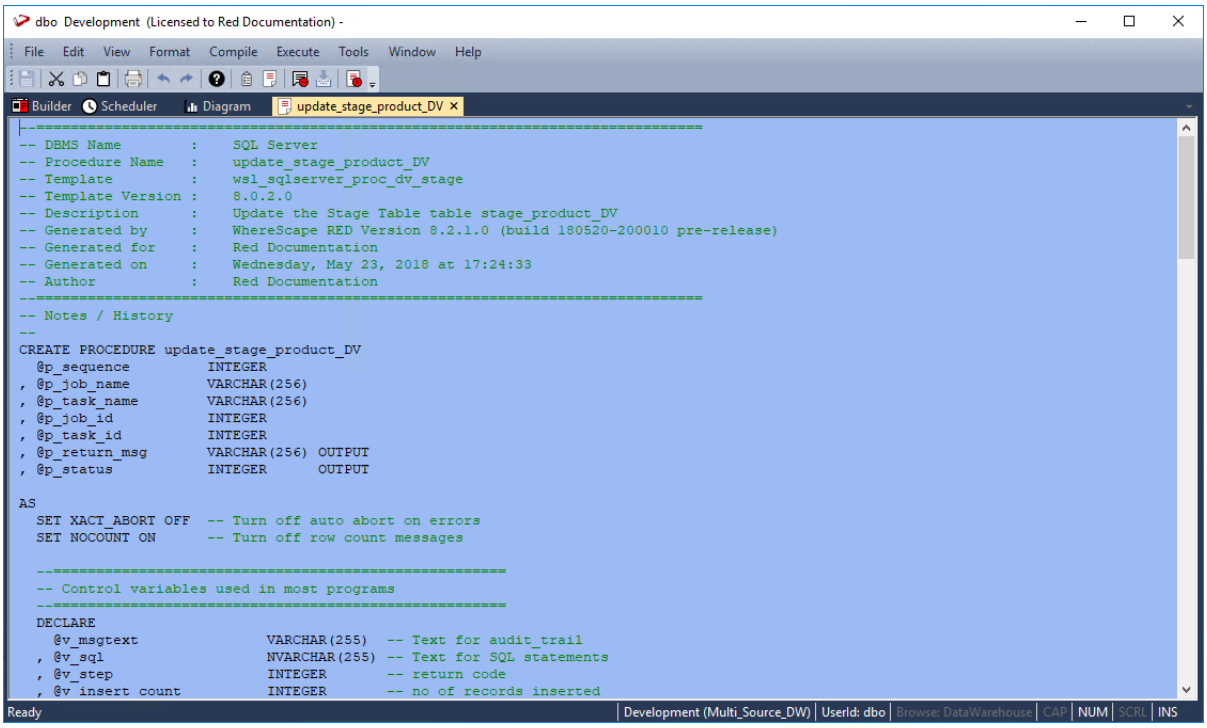


3. Click **OK** to perform the procedure generation. The **Results** pane displays confirmation that the procedure was generated.

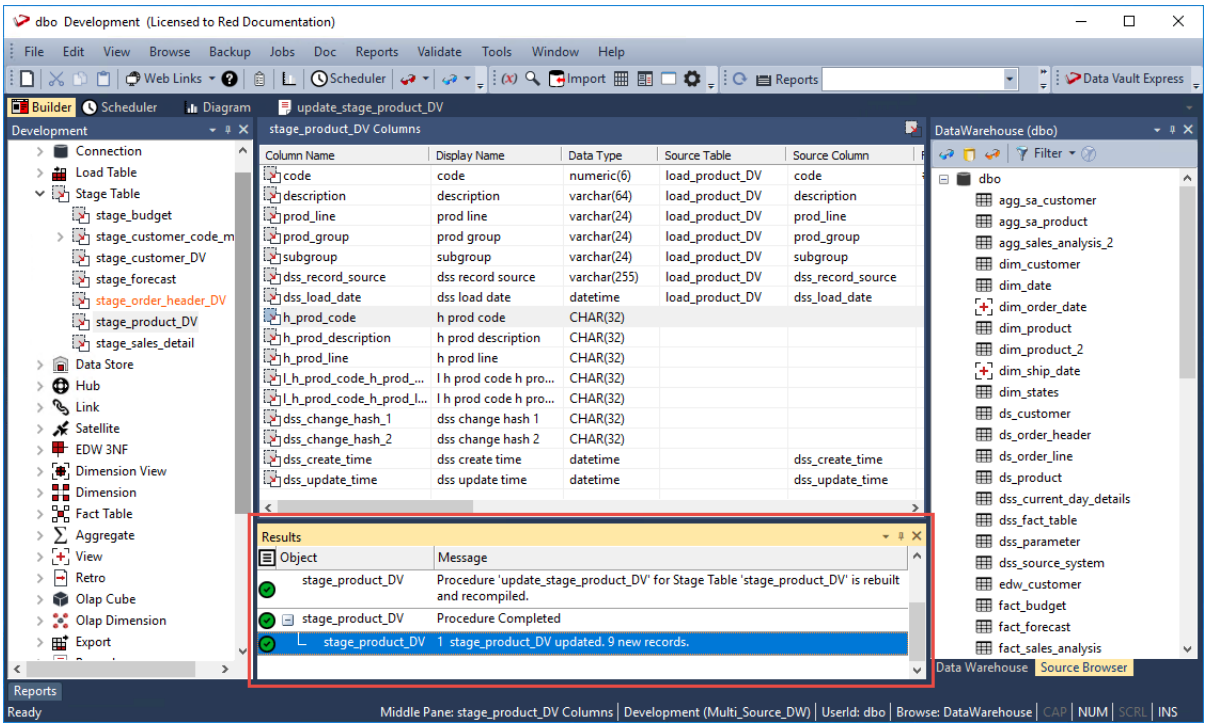


4. Right-click the **Data Vault Stage** table in the left pane, under the **Stage Table** objects list and select **Code > View update** from the context menu to view the code generated for the update procedure generated.

© Copyright, WhereScape Software Ltd All rights reserved.



5. Right-click the **Data Vault Stage** table in the left pane and select **Execute Update Procedure** from the context menu to run the procedure. The **Results** pane displays the number of records created.



© Copyright, WhereScape Software Ltd All rights reserved.

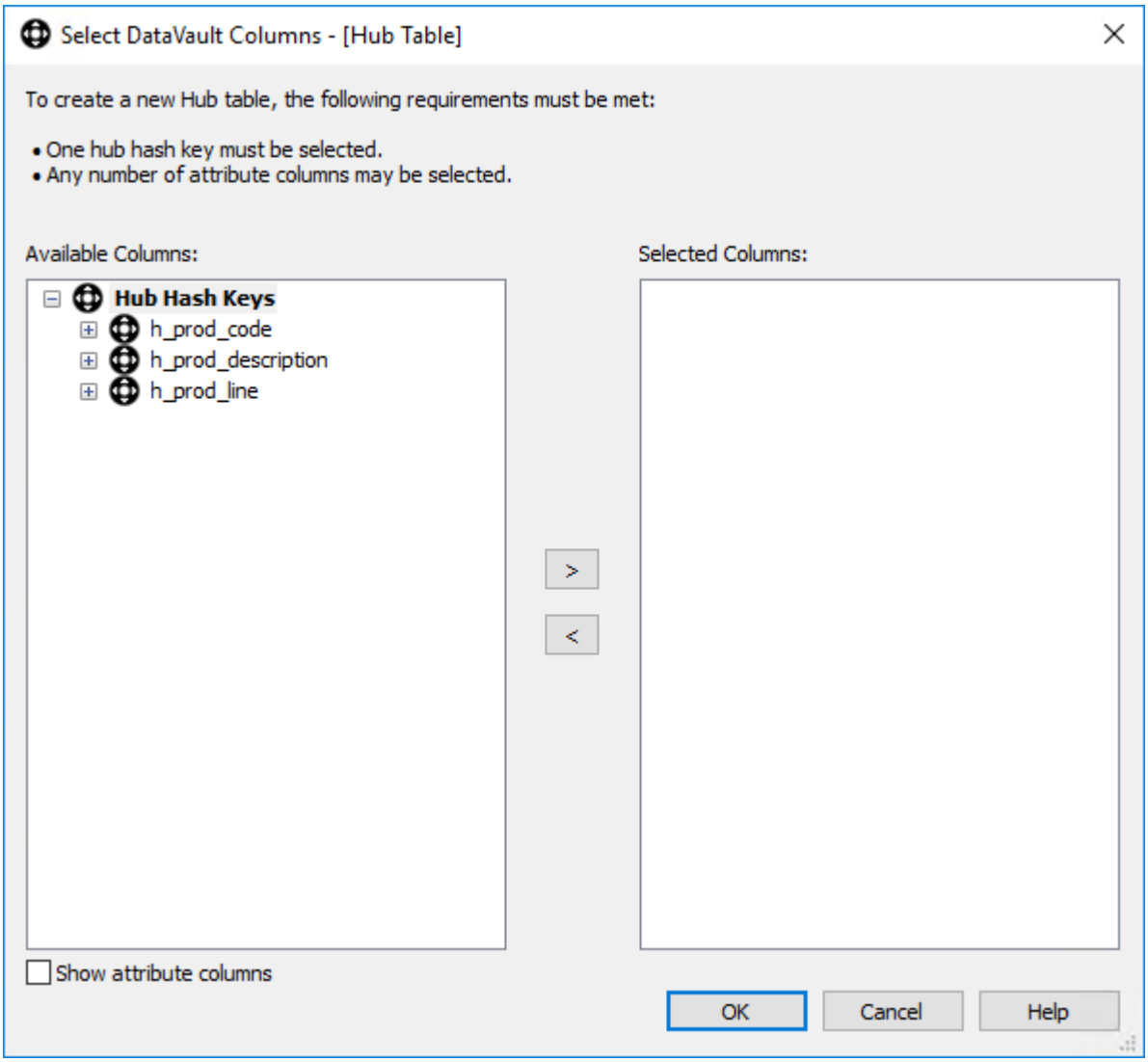
Creating the Hub, Link and Satellite Tables

After successfully creating and populating the **Data Vault Stage** table, you can now create the **Hub**, **Link** and **Satellite** tables. The **Hub**, **Link** and **Change** hash keys information stored in the **Data Vault Stage** table is used by the Wizard for building these **Data Vault** objects.

Creating the Hub table

The following describe the steps for creating a Hub table:

1. Browse to the Data Warehouse connection (click **Browse >Data Warehouse**) to create the Hub table.
2. Double-click the **Hub** object group in the left pane, the middle pane displays a list of existing Hub tables.
3. Click the source **Data Vault Stage** table from the right pane and drag it to the middle pane.
4. The Hub table creation Wizard appears and prompts you to select the Hash Keys to use from the **Available Columns** pane. The columns that comprise the selected Hash Key are displayed under the **Selected Columns** pane—these are the columns that will be populated by the Wizard in the new **Hub** table.

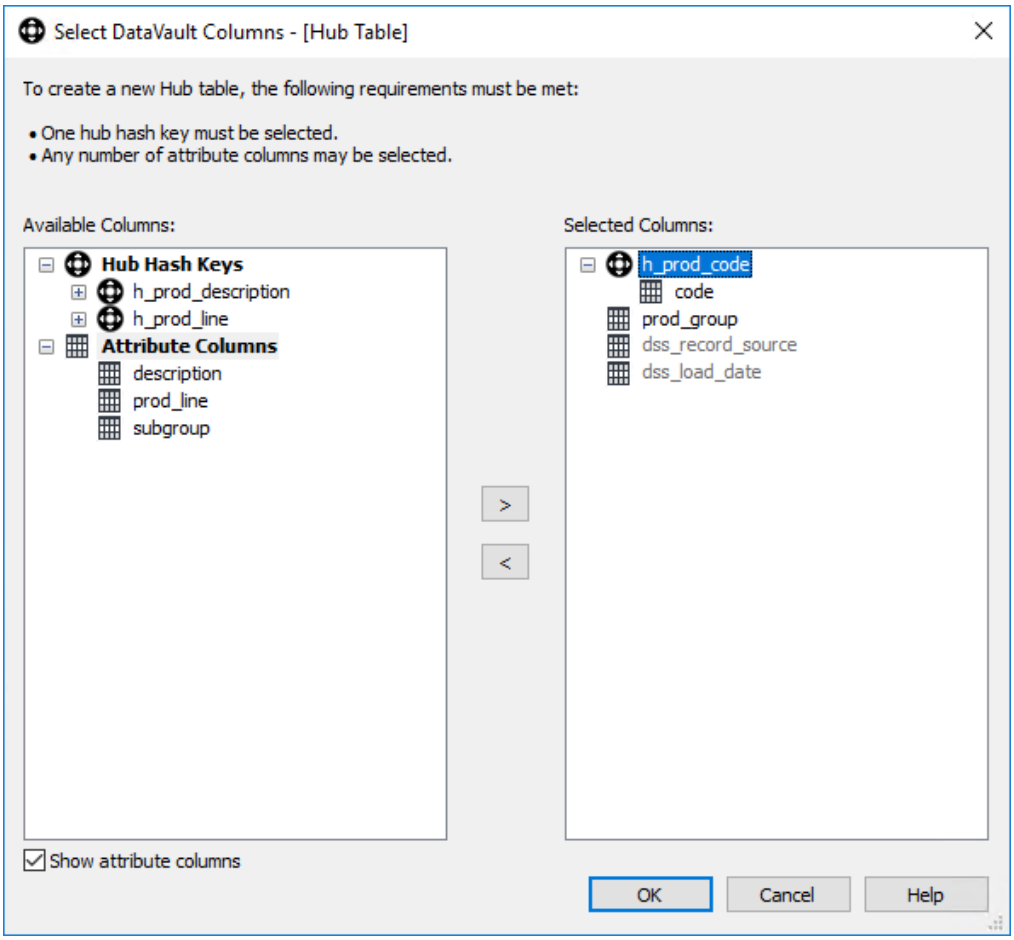


5. Select the Hash Keys you want to use from the **Available Columns** pane to see the columns that will be included in your Hub table under the **Selected Columns** pane. Click **OK** to continue.

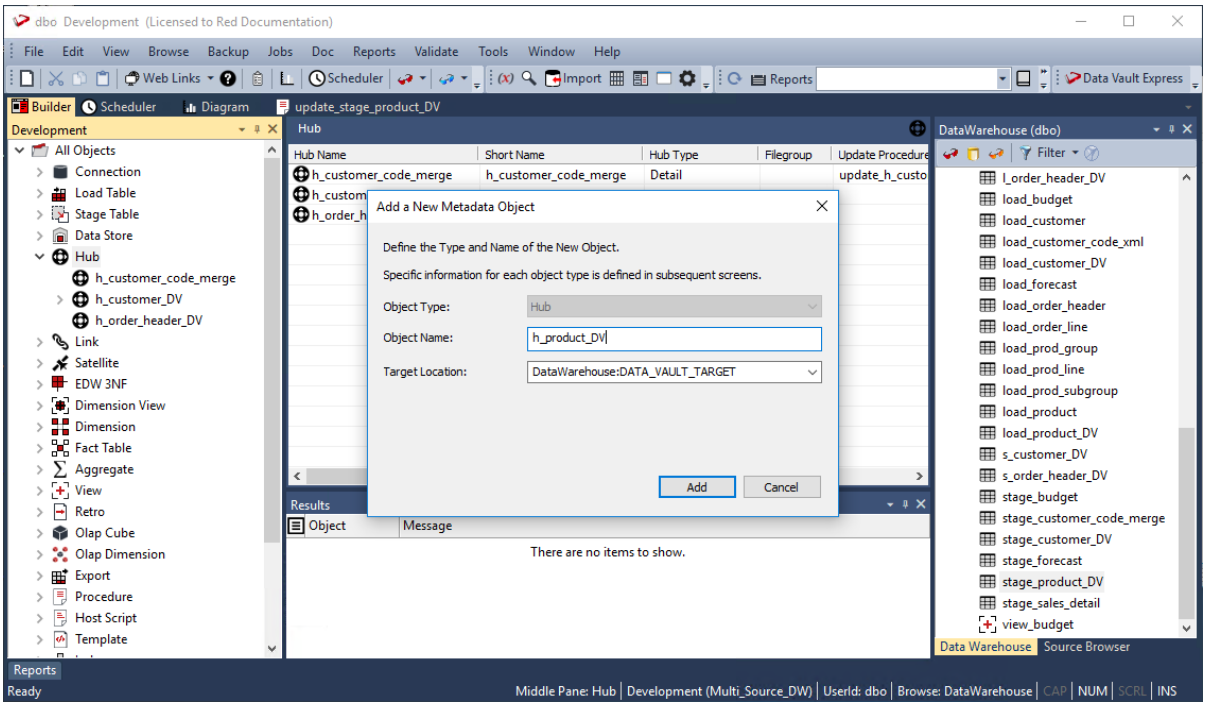
Note:

In addition to the Hash Keys, you have the option to use attribute columns to create the Hub table. Select the **Show attribute columns** check box below the **Available Columns** pane to display and select the attribute columns that will be populated by the Wizard in the new Hub table.

© Copyright, WhereScape Software Ltd All rights reserved.

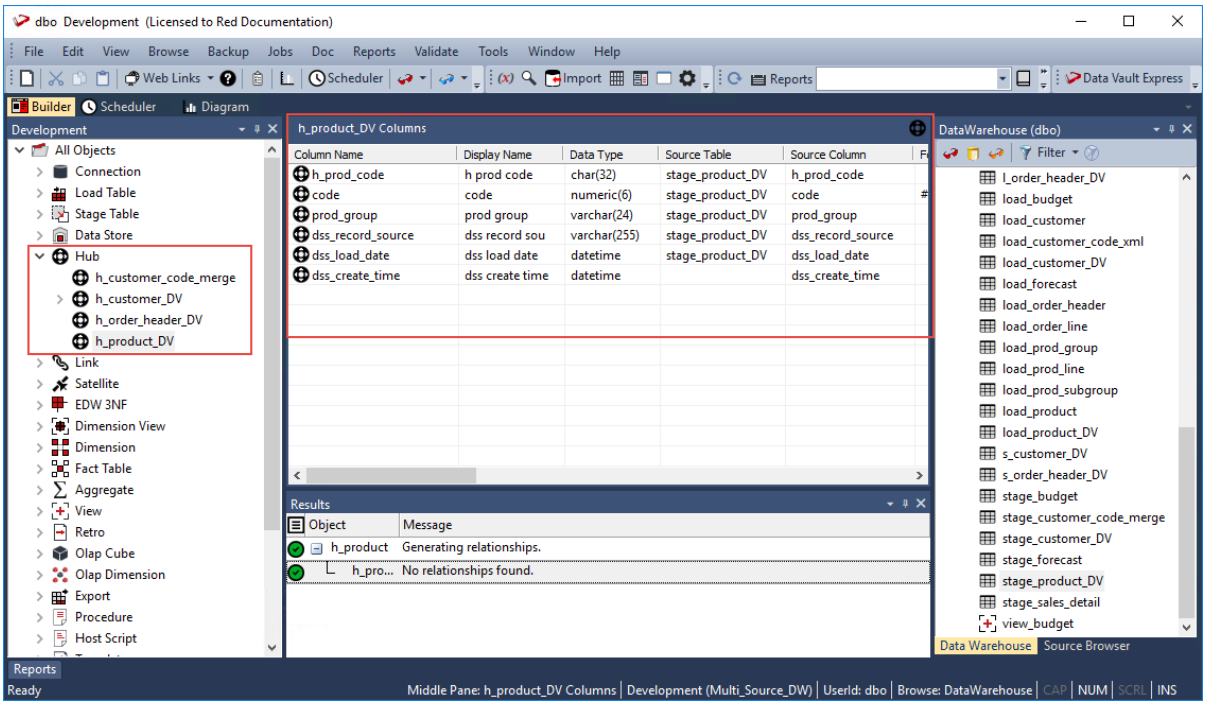


6. The **Add a New Metadata Object** screen appears and classifies the new object as a Hub table. It provides a default name based on the source **Data Vault Stage** table name. Accept the name or enter a new name for the Hub table and click **ADD** to continue.

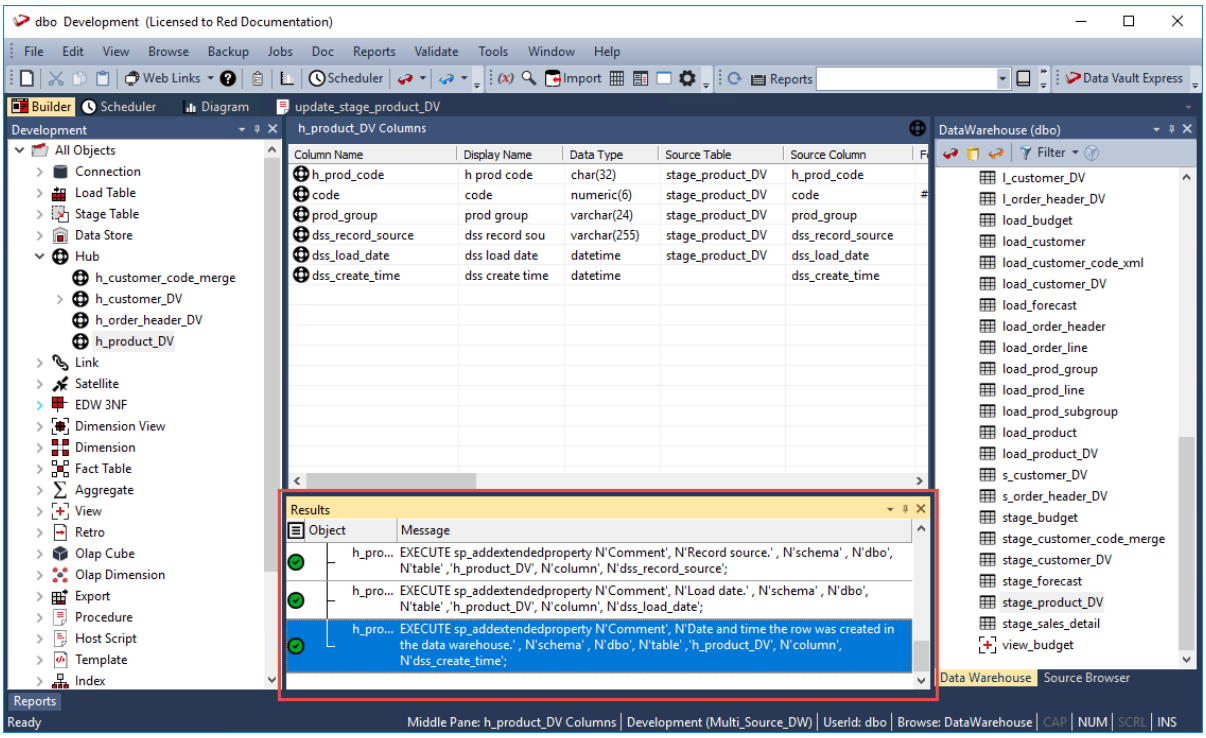


7. Click **OK** in the **Table Properties** screen to finish defining the meta data for the Hub table. The new Hub table is added to the **Hub** objects list in the left pane and the columns included in the table are listed in the middle pane.

© Copyright, WhereScape Software Ltd All rights reserved.



8. Right-click the new Hub table you defined in the left pane and select **Create (ReCreate)** from the context menu to create the table. The **Results** pane displays confirmation that the Hub table was successfully created.



Note:

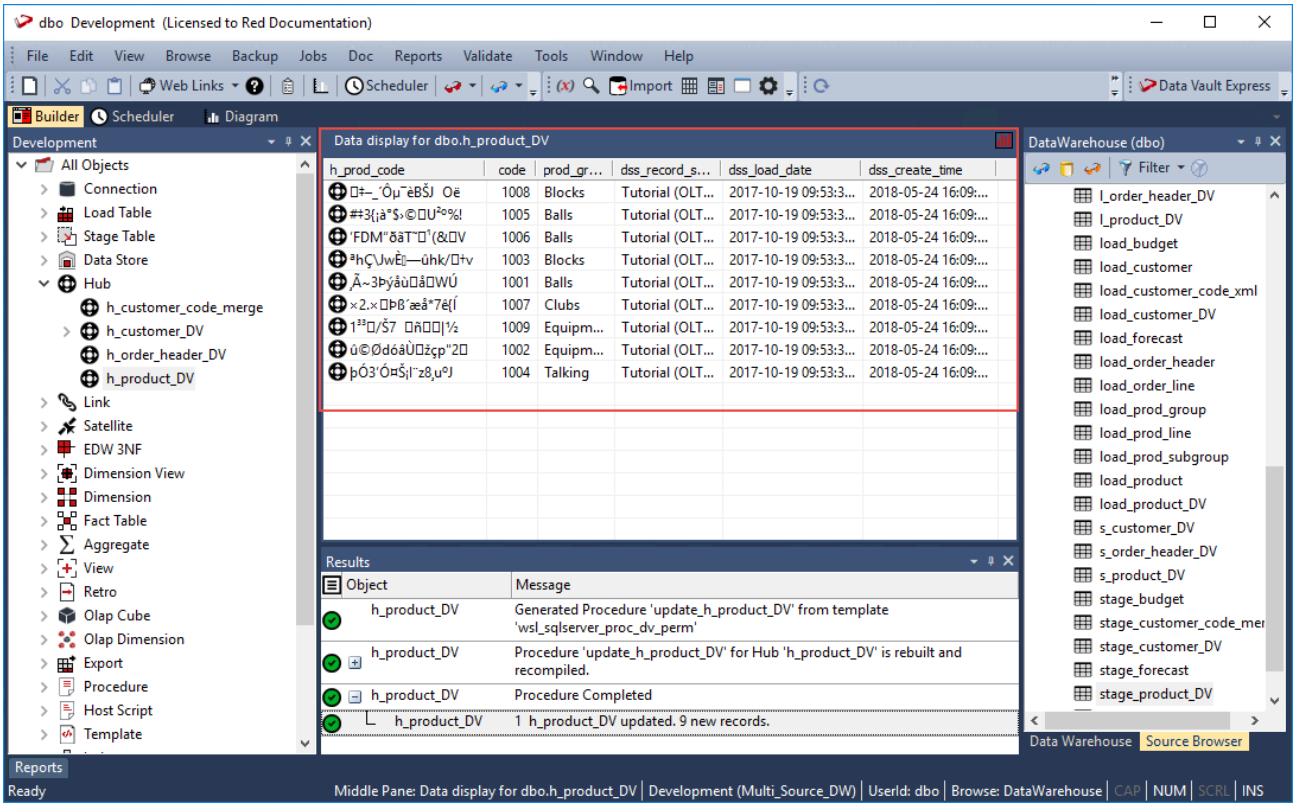
For Teradata users, click **Yes** on the Primary Index prompt and then specify **code** in the **Primary Index Columns** field of the table **Properties** screen.

After the new Hub table is defined and created, clicking the **Rebuild** button in the table **Properties** screen launches the Wizard for generating the update procedure to populate the table. This Wizard utilizes a template to create the procedure.

The detailed steps for using this Wizard is described in **Generating Update Procedures for the Hub, Link and Satellite Tables**.

© Copyright, WhereScape Software Ltd All rights reserved.

Once you have run the generated update procedures, you can view the generated Hub Hash keys, by right-clicking the new **Hub** table you created in the left pane and then selecting **Display Data** from the context menu:

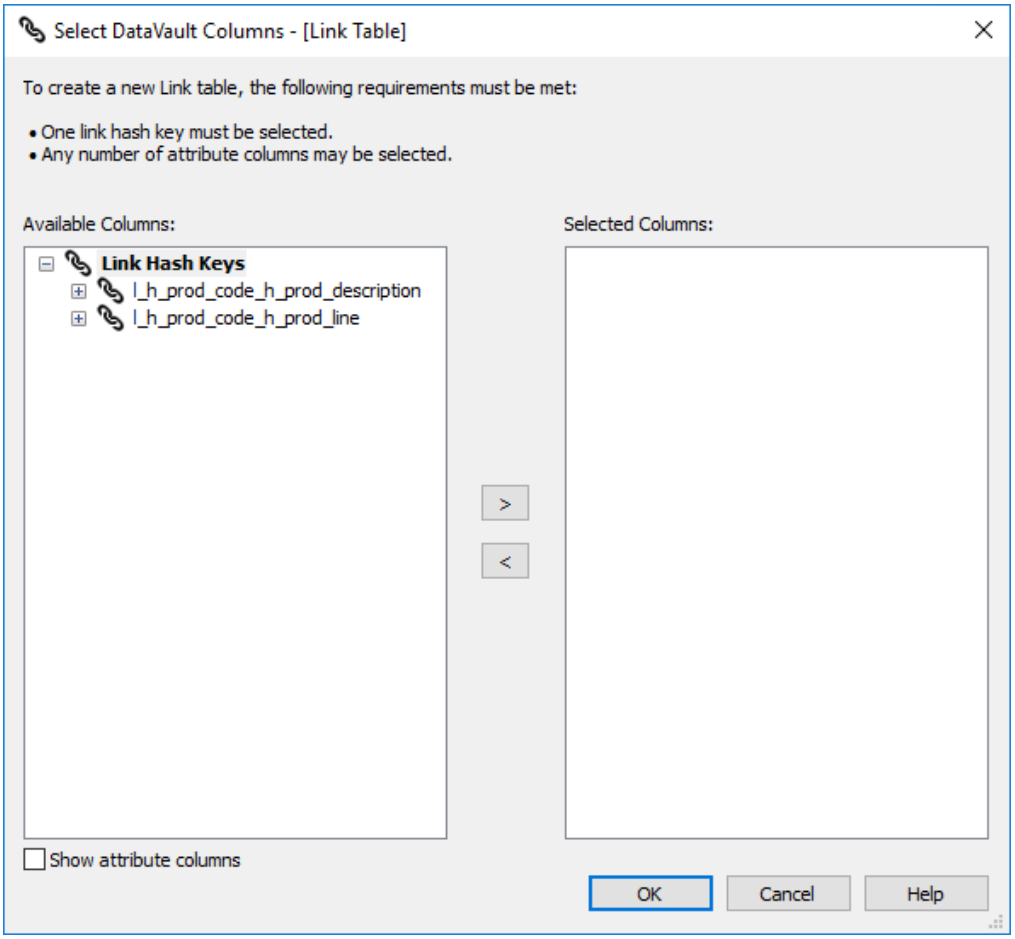


© Copyright, WhereScape Software Ltd All rights reserved.

Creating the Link table

The steps for creating the Link table is similar to the steps used in creating the Hub table:

1. Browse to the Data Warehouse connection to create the Link table.
2. Double-click the **Link** object group in the left pane, the middle pane displays a list of existing Link tables.
3. Click the source **Data Vault Stage** table from the right pane and drag it to the middle pane.
4. The Link table creation Wizard appears and prompts you to select the Hash Key to use from the **Available Columns** pane. The columns that comprise the selected Hash Key are displayed under the **Selected Columns** pane—these are the columns that will be populated by the Wizard in the new **Link** table.

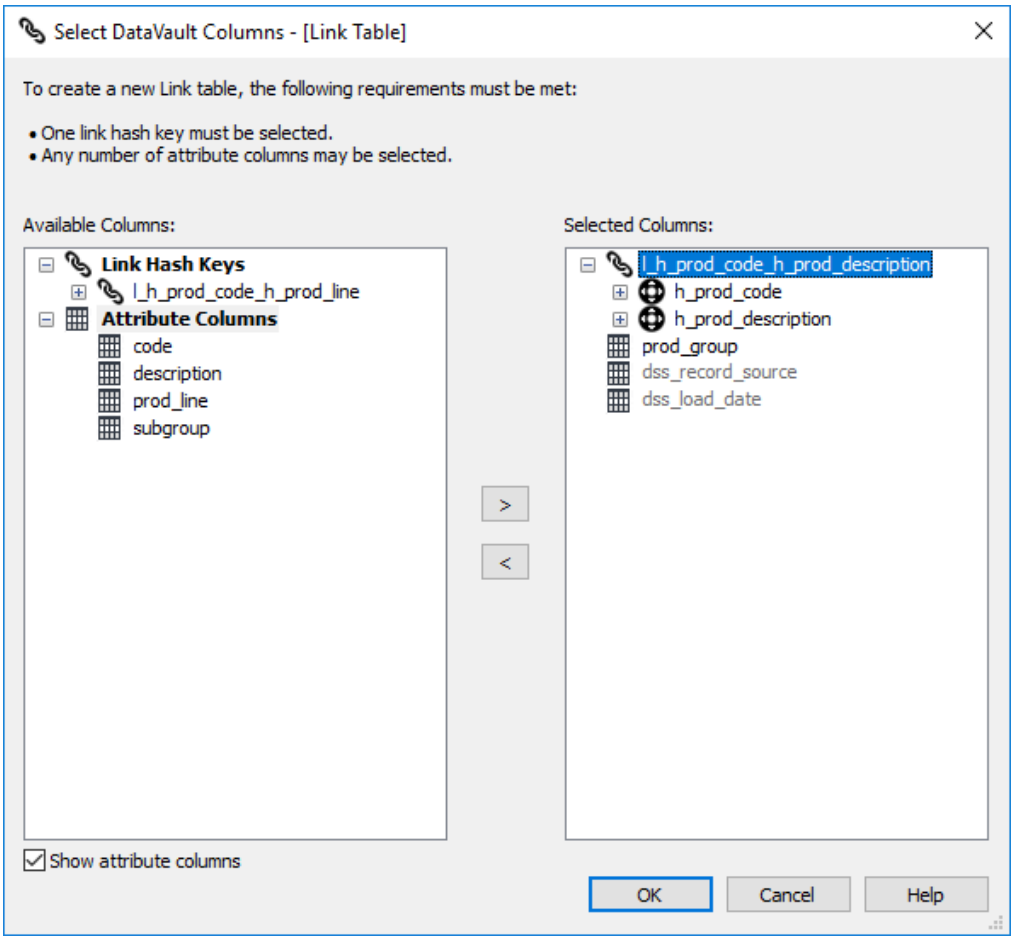


5. Select the Hash Key you want to use from the **Available Columns** pane to see the columns that will be included in your Link table under the **Selected Columns** pane. Click **OK** to continue.

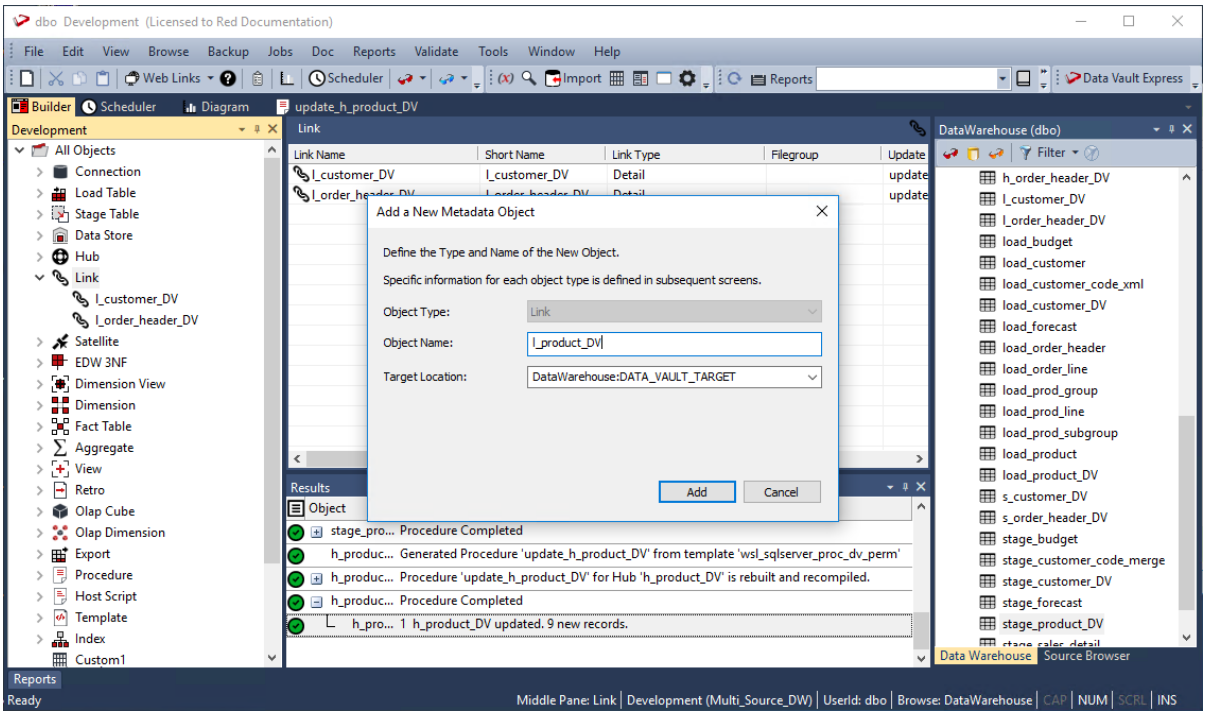
Notes

In addition to the Hash Keys, you have the option to use attribute columns to create the Link table. Select the **Show attribute columns** check box below the **Available Columns** pane to display and select the attribute columns that will be populated by the Wizard in the new Link table.

© Copyright, WhereScape Software Ltd All rights reserved.

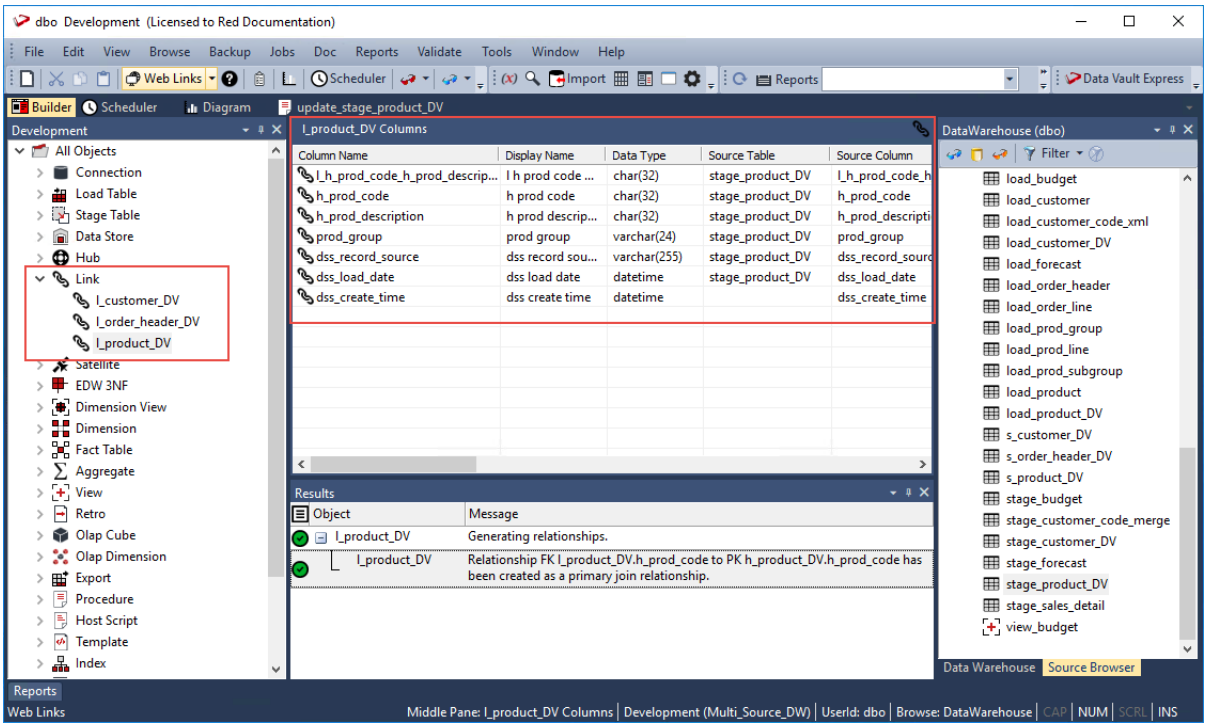


6. The **Add a New Metadata Object** screen appears and classifies the new object as a Link table. It provides a default name based on the source **Data Vault Stage** table name. Accept the name or enter a new name for the Link table and click **ADD** to continue.

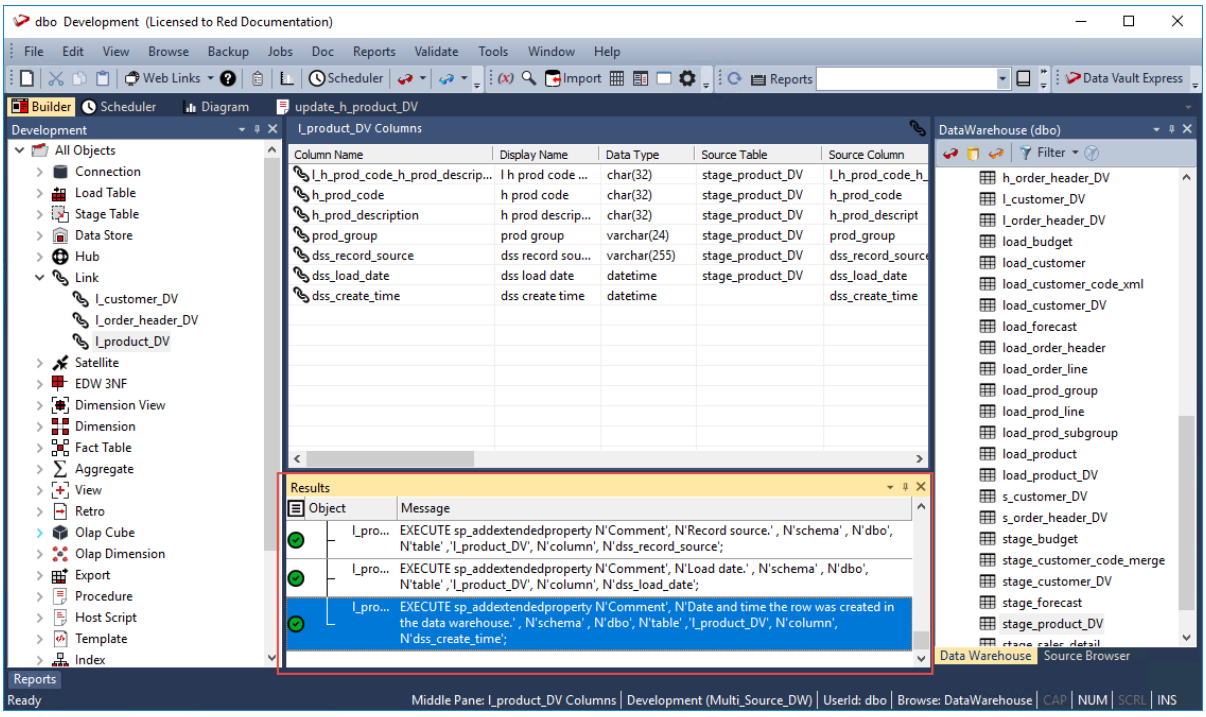


7. Click **OK** in the table **Properties** screen to finish defining the meta data for the Link table. The new Link table is added to the **Link** objects list in the left pane and the columns included in the table are listed in the middle pane.

© Copyright, WhereScape Software Ltd All rights reserved.



8. Right-click the new Link table you defined in the left pane and select **Create (ReCreate)** from the context menu to create the table. The **Results** pane displays confirmation that the Link table was successfully created.



Note

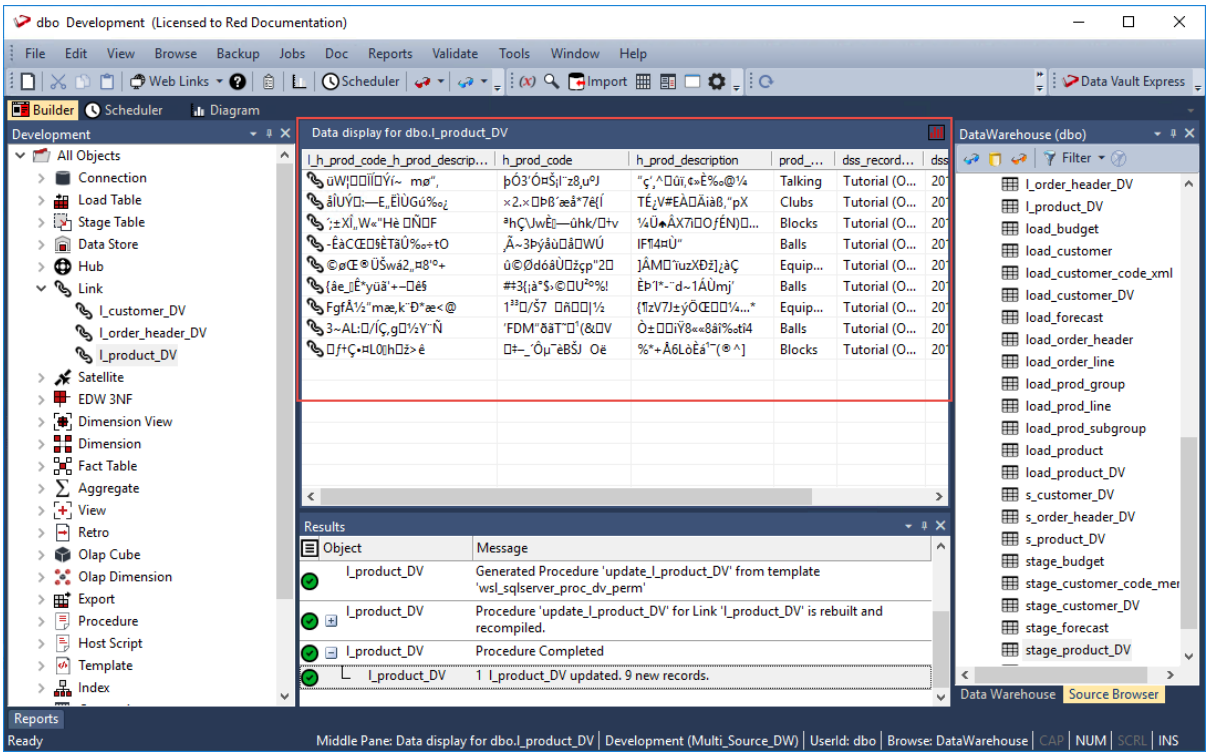
For Teradata users, click **Yes** in the Primary Index prompt and then specify **code** in the **Primary Index Columns** field of the table **Properties** screen.

After the new Link table is defined and created, clicking the **Rebuild** button in the table **Properties** screen launches the Wizard for generating the update procedure to populate the table. This Wizard utilizes a template to create the procedure.

© Copyright, WhereScape Software Ltd All rights reserved.

The detailed steps for using this Wizard is described in **Generating Update Procedures for the Hub, Link and Satellite Tables**.

Once you have run the generated update procedures, you can view the generated Link Hash keys, by right-clicking the new Link table you created in the left pane and then selecting **Display Data** from the context menu:

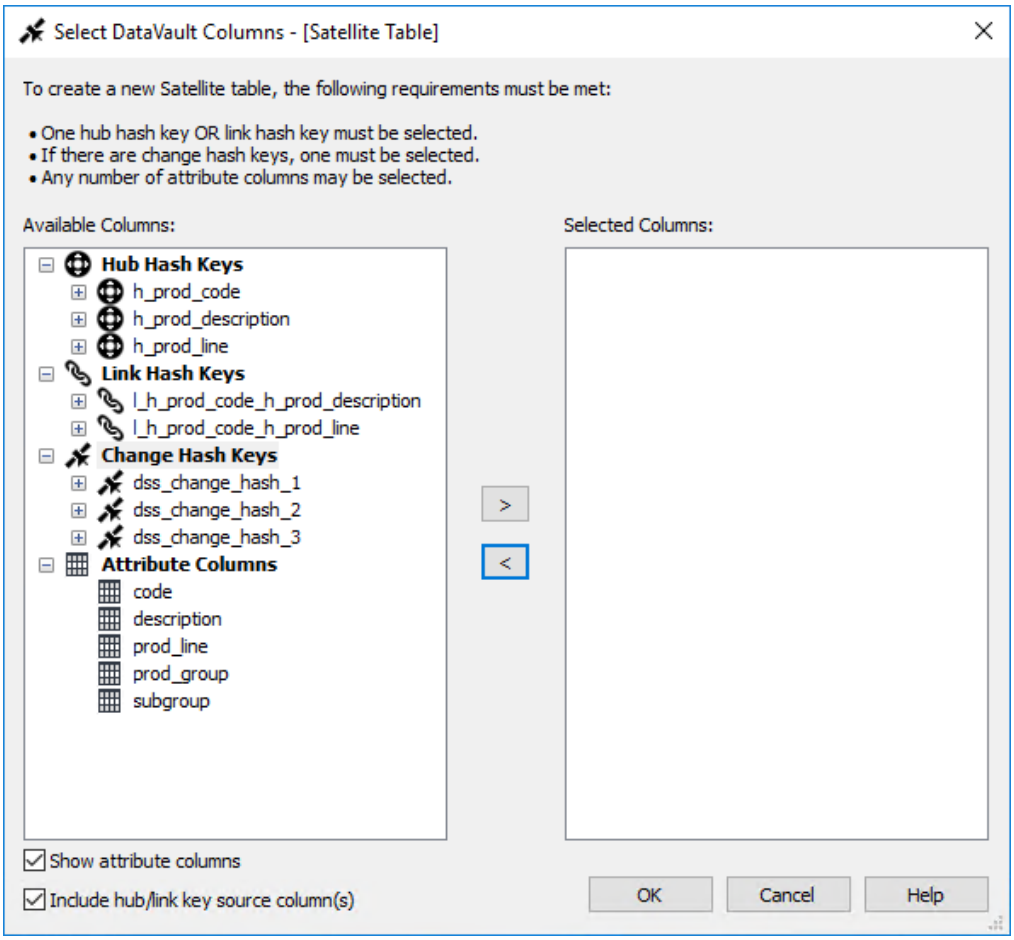


Creating the Satellite table

The following describe the steps for creating a Satellite table:

1. Browse to the Data Warehouse connection to create the Satellite tables.
2. Double-click the **Satellite** object group in the left pane, the middle pane displays a list of existing Satellite tables.
3. Click the source **Data Vault Stage** table from the right pane and drag it to the middle pane.
4. The Satellite table creation Wizard appears and prompts you to select the Hash Key to use from the **Available Hash Keys** pane. The columns that comprise the selected Hash Key are displayed under the **Selected Columns** pane—these are the columns that will be populated by the Wizard in the new Satellite table.

© Copyright, WhereScape Software Ltd All rights reserved.

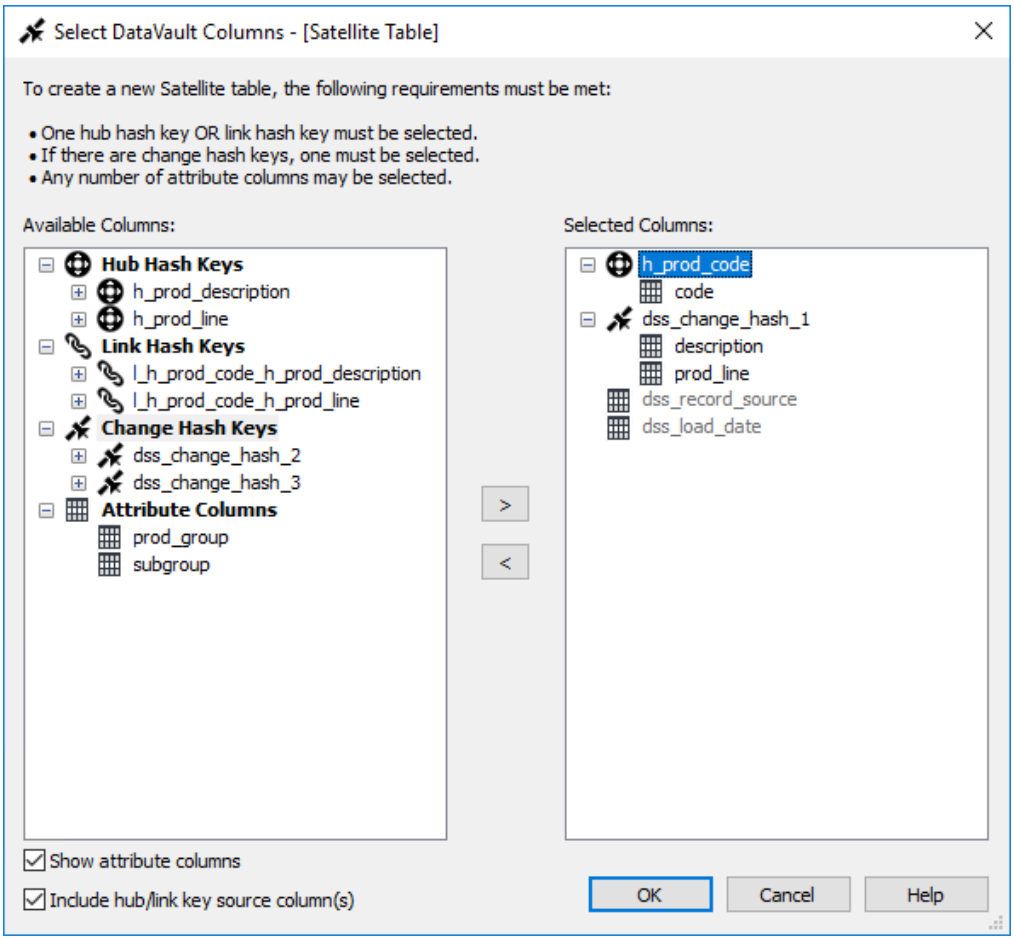


5. Select the Hash Key you want to use from the **Available Hash Keys** pane to see the columns that will be included in your Satellite table under the **Selected Columns** pane. Click **OK** to continue.

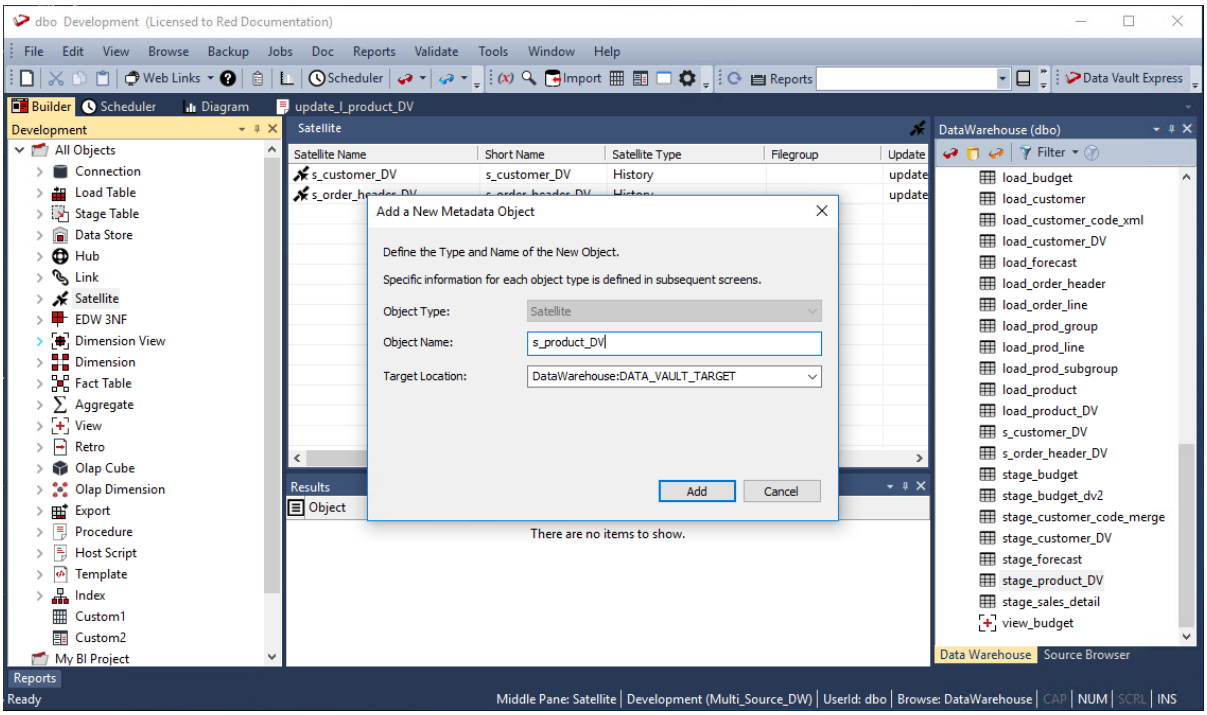
Notes

1. In addition to the Hash Keys, you have the option to use attribute columns to create the Satellite table. Select the **Show attribute columns** check box below the **Available Columns** pane to display and select the attribute columns that will be populated by the Wizard in the new Satellite table.
2. Selecting the **Include hub/link key source columns** check box will include the hub hash key source column when the Satellite table is created.

© Copyright, WhereScape Software Ltd All rights reserved.

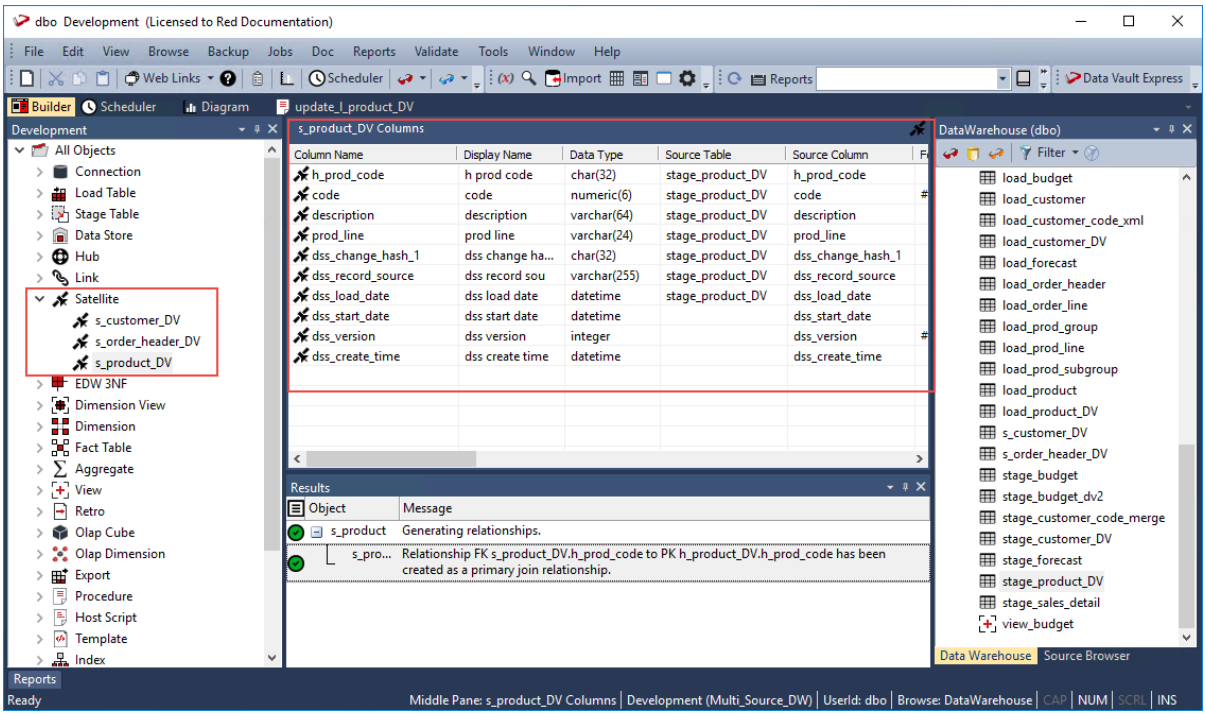


6. The **Add a New Metadata Object** screen appears and classifies the new object as a Satellite table. It provides a default name based on the source **Data Vault Stage** table name. Accept the name or enter a new name for the Satellite table and click **ADD** to continue.

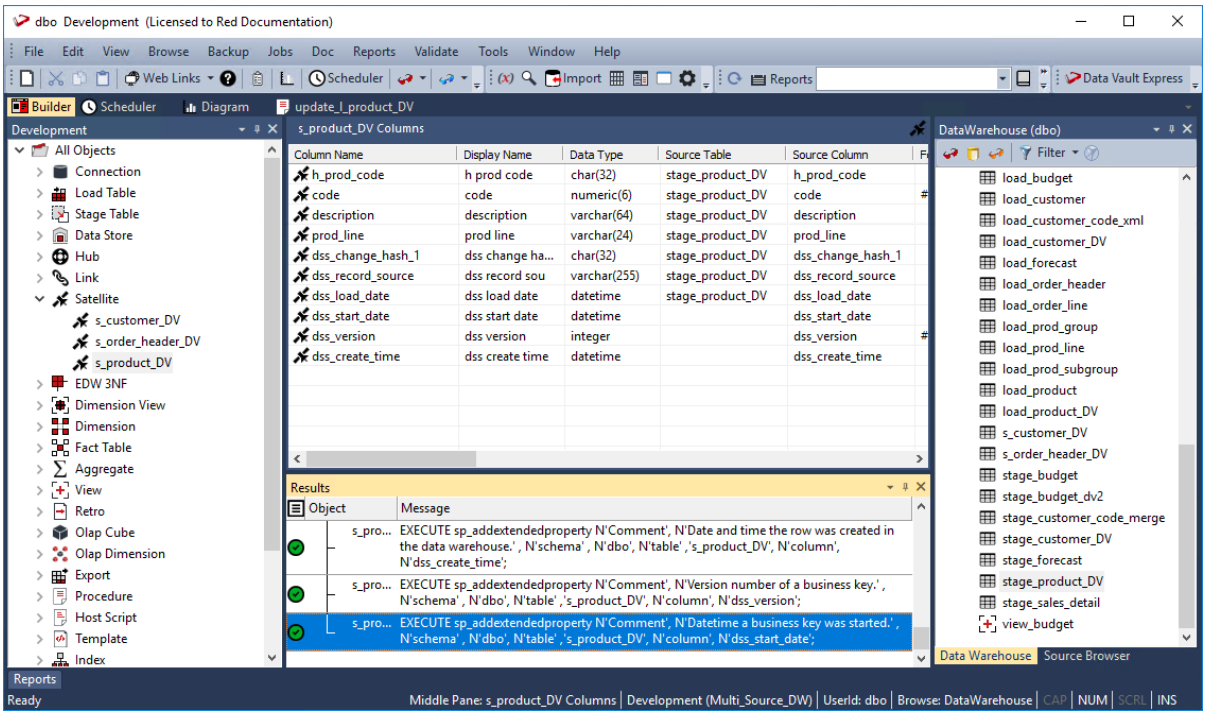


7. Click **OK** in the table **Properties** screen to finish defining the meta data for the Satellite table. The new Satellite table is added to the **Satellite** objects list in the left pane and the columns included in the table are listed in the middle pane.

© Copyright, WhereScape Software Ltd All rights reserved.



8. Right-click the new Satellite table you defined in the left pane and select **Create (ReCreate)** from the context menu to create the table. The **Results** pane displays confirmation that the Satellite table was successfully created.



Note

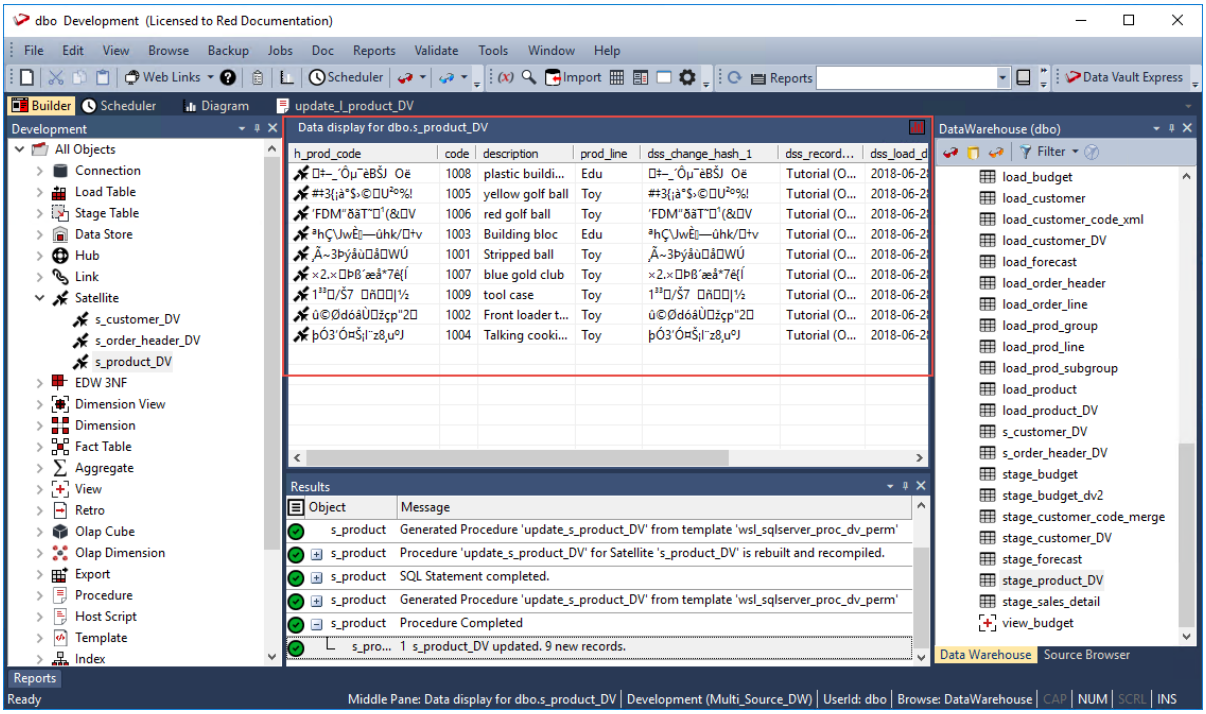
For Teradata users, click **Yes** on the Primary Index prompt and then specify **code** on the **Primary Index Columns** field of the table **Properties** screen.

After the new Satellite table is defined and created, clicking the **Rebuild** button in the **Table Properties** screen launches the Wizard for generating the update procedure to populate the table. This Wizard utilizes a template to create the procedure.

The detailed steps for using this Wizard is described in **Generating Update Procedures for the Hub, Link and Satellite Tables**.

© Copyright, WhereScape Software Ltd All rights reserved.

Once you have run the generated update procedures, you can view the generated Satellite Hash keys, by right-clicking the new Satellite table you created in the left pane and then selecting **Display Data** from the context menu:



© Copyright, WhereScape Software Ltd All rights reserved.

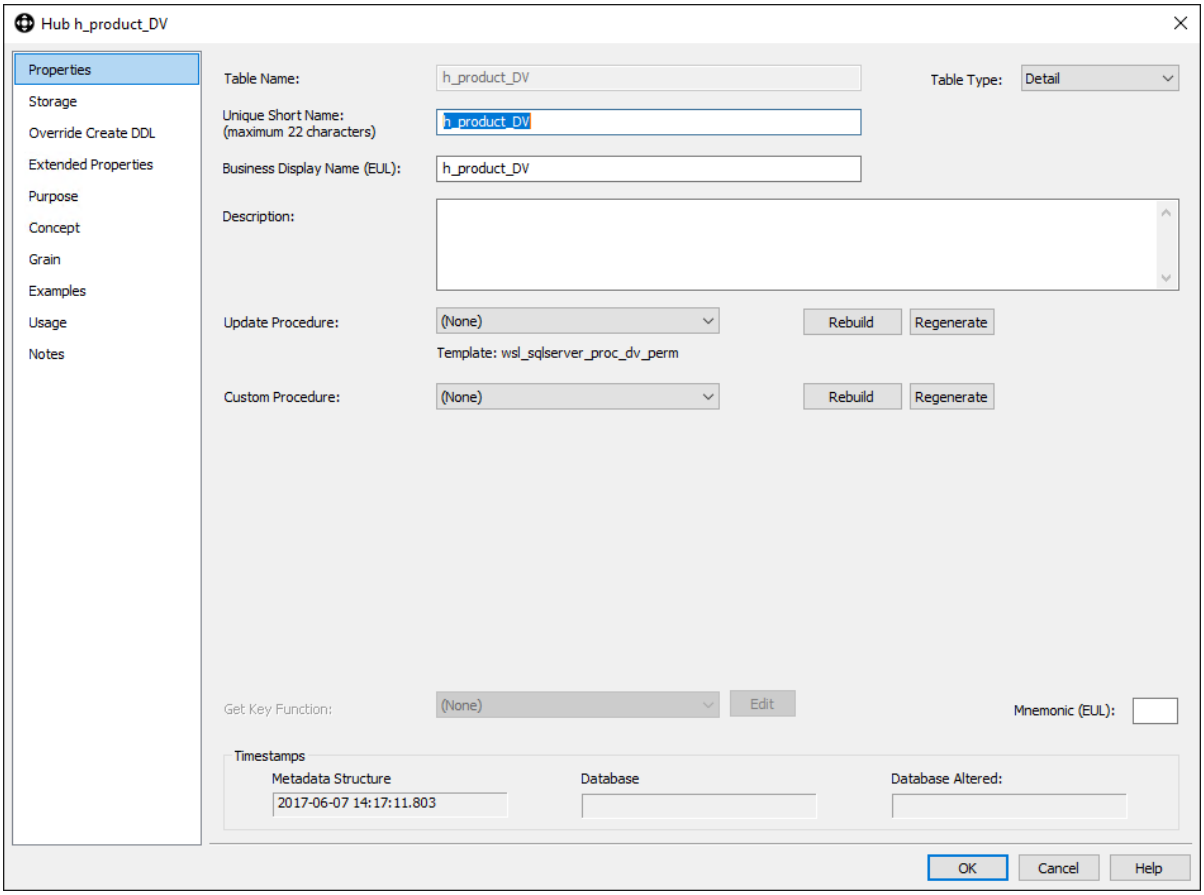
Generating Update Procedures for Hub, Link and Satellite Tables

The following describe the steps for generating update procedures via a template.

Hub table

After successfully defining and creating the Hub table, you can generate the update procedure via a template to populate the table.

1. Click the **Rebuild** button in the table **Properties** screen to launch the procedure generation Wizard to populate the table.

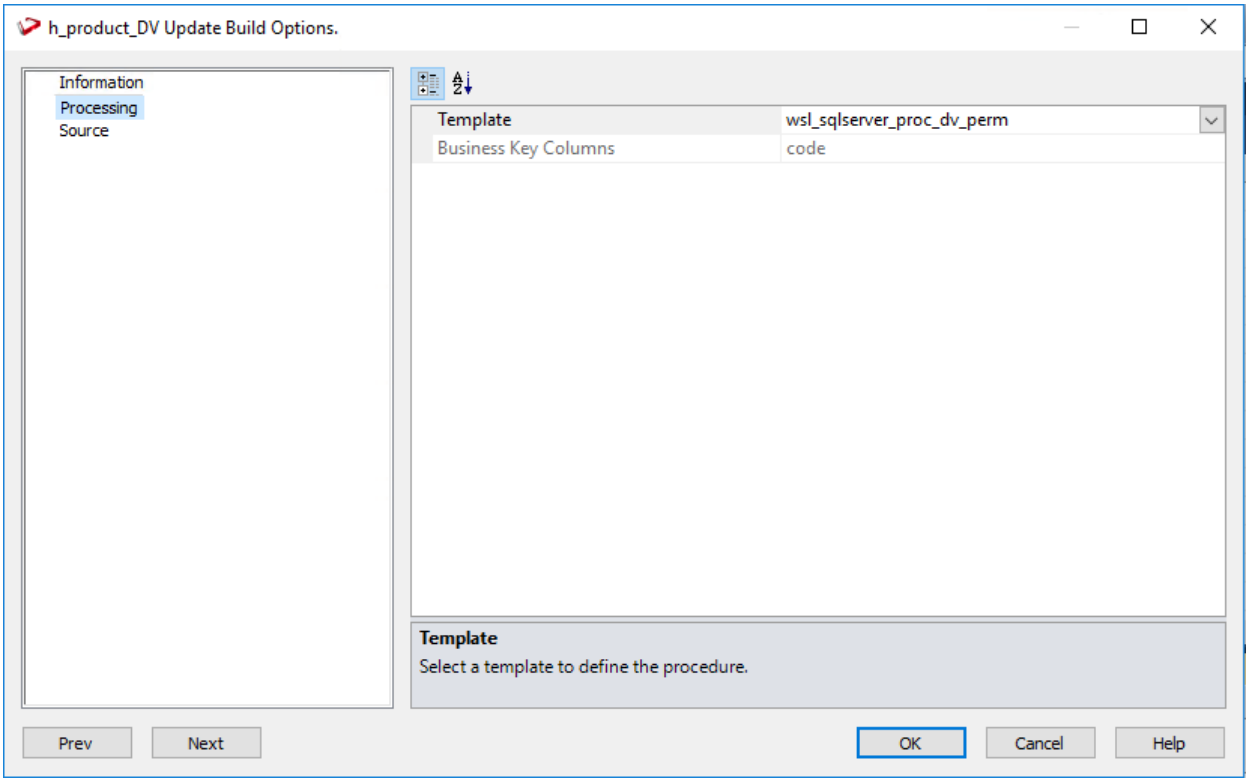


Note

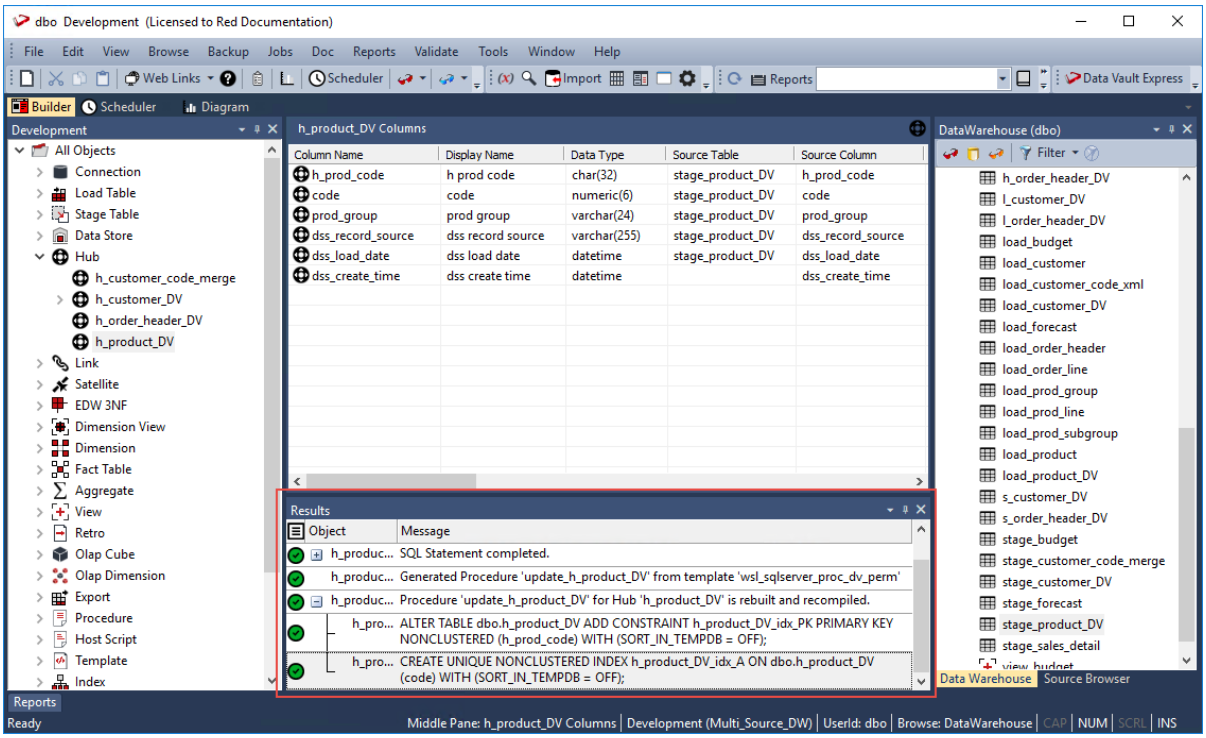
RED displays the name of the previously used update procedure template by default, below the **Update Procedure** drop-down.

© Copyright, WhereScape Software Ltd All rights reserved.

2. On the **Processing** tab of **Table Update Build Options** screen, select the template to use from the **Template** drop-down or use the previous update procedure template.

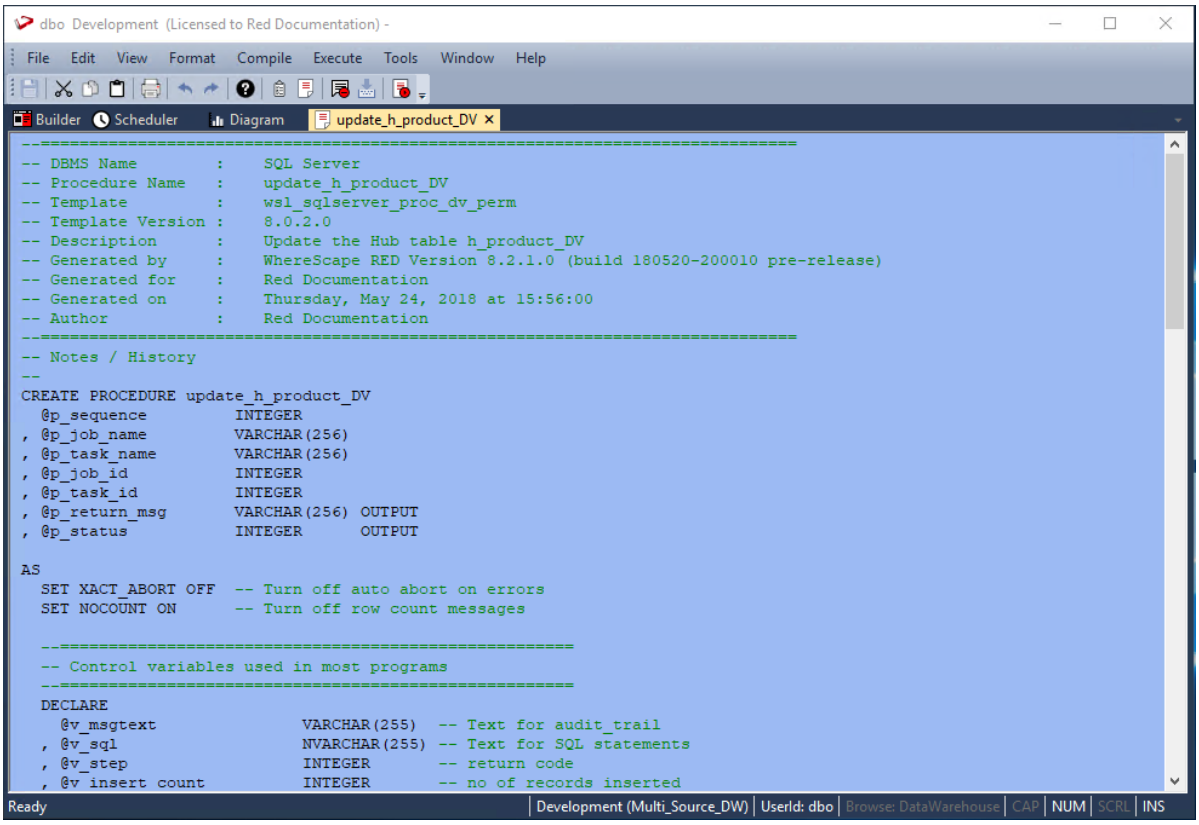


3. Click **OK** to proceed with the procedure generation. The **Results** pane displays confirmation that the procedure was generated.

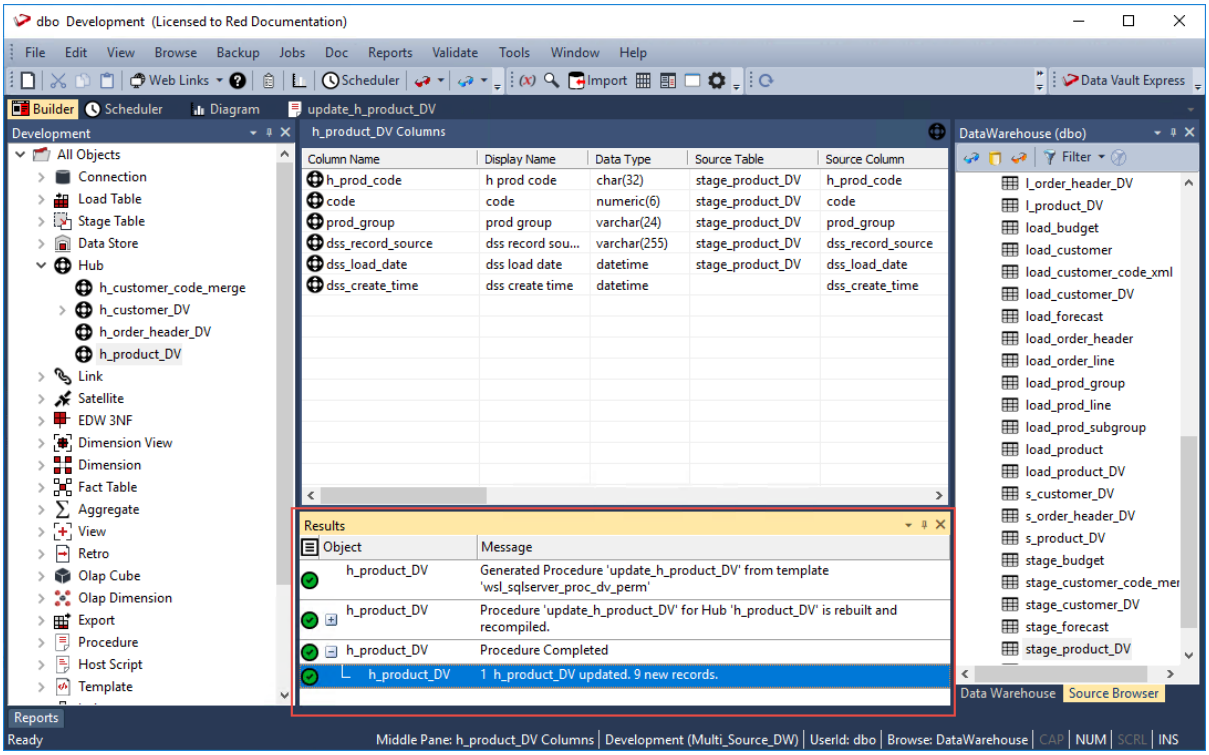


4. Right-click the Hub table in the left pane, under the **Hub** objects list and select **Code > View update** from the context menu to view the contents of the update procedure generated.

© Copyright, WhereScape Software Ltd All rights reserved.



5. Right-click the Hub table in the left pane and select **Execute Update Procedure** from the context menu to run the procedure. The **Results** pane displays the number of records created.



Link and Satellite Tables

Follow the same steps described above to create and execute the update procedures for the Link and Satellite tables, via a template to populate the tables.

© Copyright, WhereScape Software Ltd All rights reserved.

Changing the Data Vault Hash Key Function in WhereScape RED 6.9.1.0 and above

WhereScape RED ships templates (refer to **Data Vault Templates**) to generate the procedures for **Data Vault** objects if the customer's license includes **Data Vault** object support and for database platforms SQL Server, Oracle or Teradata.

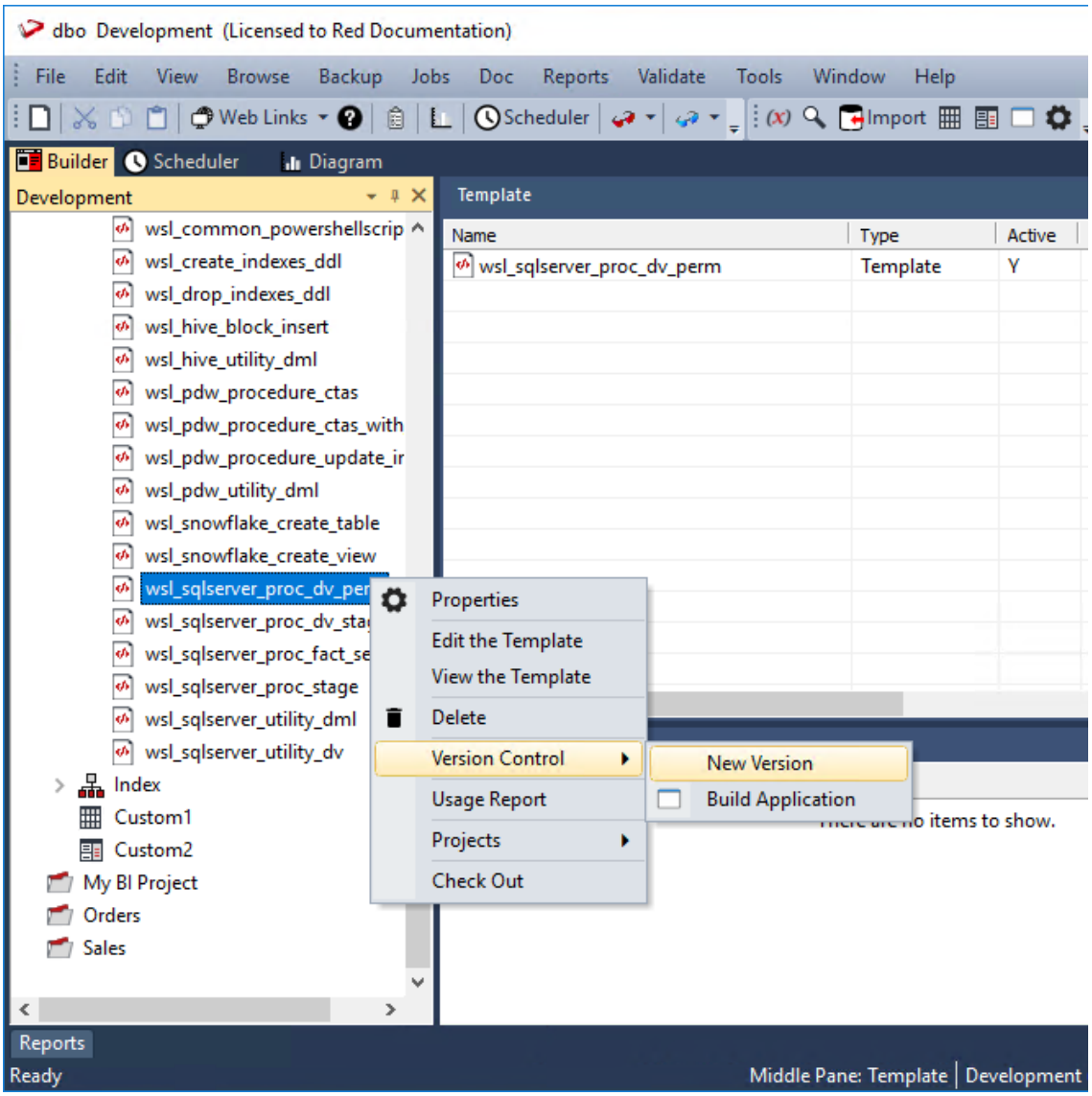
Included in the **Data Vault** utility (wsl_<database>_utility_dv) template is an **initial Hash key function**. This Hash key function must be evaluated before deployment, and if an alternate function is preferable for the customer's operating environment, then this function must be amended before Hash keys are created in a production environment.

The following outline the suggested steps to amend the shipped Hash key function, if required.

WhereScape **Data Vault** templates are Read-Only (to differentiate from custom templates), the first step is to make a copy of each of the three templates:

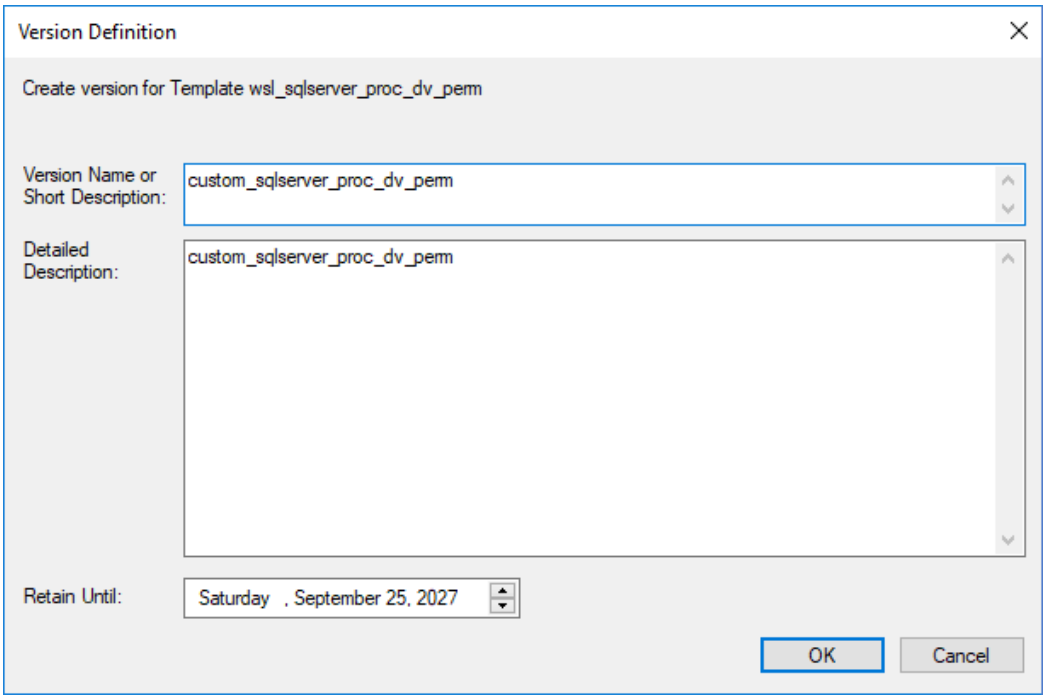
- wsl_<database>_proc_dv_perm
- wsl_<database>_proc_dv_stage
- wsl_<database>_utility_dv

1. Right-click the template from the **Template** objects list and select **Version Control > New Version** from the context menu.

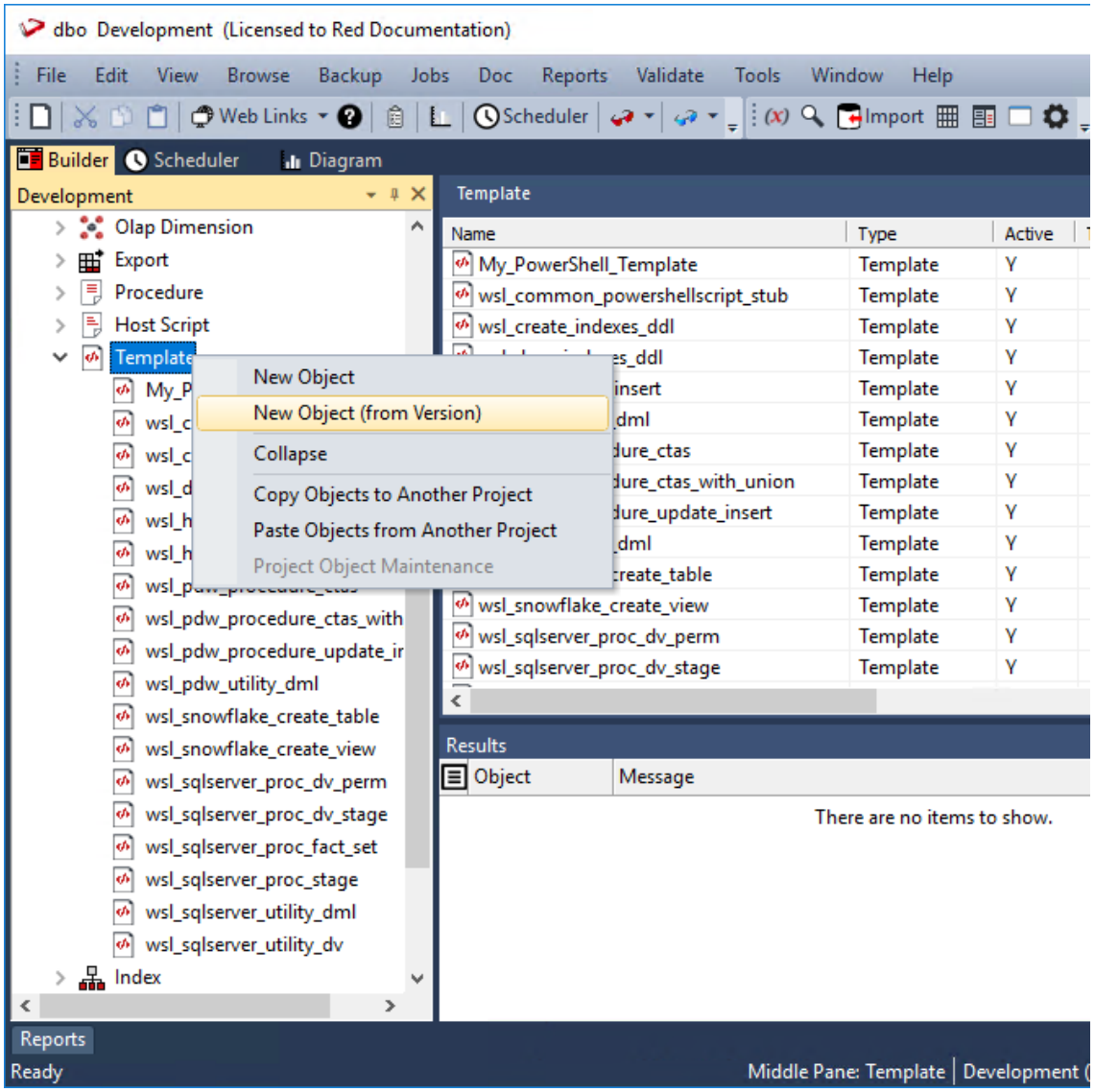


2. Enter the version name and other details on the **Version Definition** window and then click **OK** to save the new version of the template.

© Copyright, WhereScape Software Ltd All rights reserved.

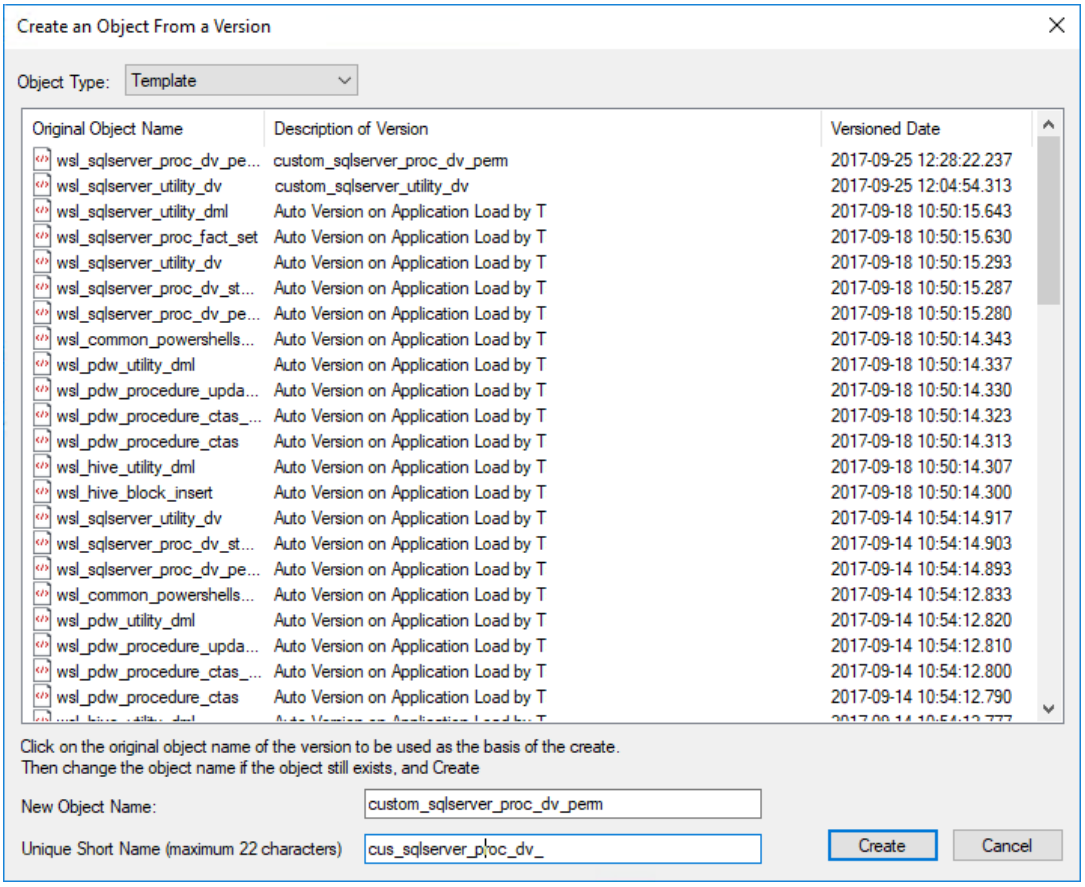


3. Right-click the **Template** object type and select **New Object (from version)** from the context menu.



© Copyright, WhereScape Software Ltd All rights reserved.

4. Select the new version of the template you created from the list in the **Create an Object From a Version** window and then enter the new object name.



5. Click **Create** to create the new Template object.
6. Follow the same steps to create the other new templates with unique names.
7. Modify the replacement utility template with the new function in this section:

```
{%- set hashFunctionPatBeg  = "CAST(HASHBYTES('md5',"       -%}

{%- set hashFunctionPatEnd  = ") AS CHAR(32))"  -%}

{%- set hashColTransPattern = "COALESCE(CAST([SRCCOL] AS VARCHAR(MAX)),'null')"  -%}

{%- set hashConcatPattern   = " +'||'+"         -%}
```

   In each of the addHubHashKey, addLinkHashKey and addChangeHashKey macros.

8. In the two replacement proc_dv templates, change the import command to reference the new utility template:

```
{% import "custom_sqlserver_utility_dv" %}
```

9. Update existing **Data Vault Stage** table objects to reference the replacement proc_dv_stage template (the **Usage Report** on the context menu of the old template objects can help locate these).
10. After successfully testing the new templates, it is recommended to delete the original **WSL Data Vault** templates to avoid confusion.

Note

The above workflow does not provide allowance for the migration of any existing **Data Vault** records, created with a previous **Data Vault** function. This workflow can be used for new **Data Vault** implementations. In an existing **Data Vault** system, changing the Hash key function requires migration of existing Hash keys which is not provided in this workflow.

© Copyright, WhereScape Software Ltd All rights reserved.

Custom Objects

Custom1 and **Custom2** objects are user defined objects. These Object types can be renamed in the **Tools > Options > Object Types > Object Names** menu.

A Custom object license is required for these object types.

Custom objects have the same options and properties as EDW 3NF tables. Refer to **EDW 3NF Tables** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

Fact Tables

A Fact table is normally defined, for our purposes, as a table with facts (measures) and dimensional keys that allow the linking of multiple dimensions. It is normally illustrated in the form of a Star Schema with the central Fact table and the outlying dimensions.

The ultimate goal of the Fact table is to provide business information to the end user community. In many cases, different types of Fact tables are required to address different end user requirements. For simplicity, the different types of fact table are grouped together under the following headings:

- **Detail**
- **Rollup**
- **KPI**
- **Snapshot** (essentially a type of Rollup fact table)
- **Slowly Changing**
- **Partitioned**

Detail Fact Tables

A detail fact table is normally a transactional table that represents the business data at its lowest level of granularity. In many ways, these tables reflect the business processes within the organization.
These Fact tables are usually large and are focused on a specified analysis area.

There may be quite a large number of Detail Fact tables in a data warehouse implementation, of which only a few are used on any regular basis by the end user community. The disadvantage of such fact tables is that they provide isolated pools of information. Although joined by conformed dimensions, it is still often difficult to answer queries across the various analysis areas.
They do however provide the ultimate drill down for all information and also the platform on which to build higher level and more complex fact tables. In terms of the time dimension detail fact tables are typically at a daily or even hourly granular level.

An example of a Detail Fact table may be the sales, or orders Fact tables that have a daily granularity, and show all sales by product, by customer etc. on a given day.

Creating Detail Fact Tables

A Detail Fact table is typically created by dragging a staging table onto a fact table list. Once released, the dialog to create the fact table will commence.

When a staging table is dragged into the drop target pane (middle pane), all Fact tables are by default **Detail Fact tables**.

If manually creating a table, then the table type can be selected from the Properties window of the Fact table.

Detail Fact Columns

The columns for a Detail Fact table are typically those of the staging table. Such Fact tables typically contain a wide range of measures and attributes as well as the business keys used to look up the artificial dimension keys.

These business keys should be included whenever possible as they provide a means of rebuilding the dimensional link. If size prohibits their inclusion it is probably be necessary to backup or archive all source data to ensure that the Fact table can be rebuilt.

These Fact tables normally contain a large number of attributes such as dates, which have not been converted to dimensions. Also contained would be information, such as order numbers, invoice numbers etc.

© Copyright, WhereScape Software Ltd All rights reserved.

Generating the Detail Fact Update Procedure

Once a detail fact table has been defined in the metadata and created in the database, an update procedure can be generated to handle the update of the fact table.

Notes

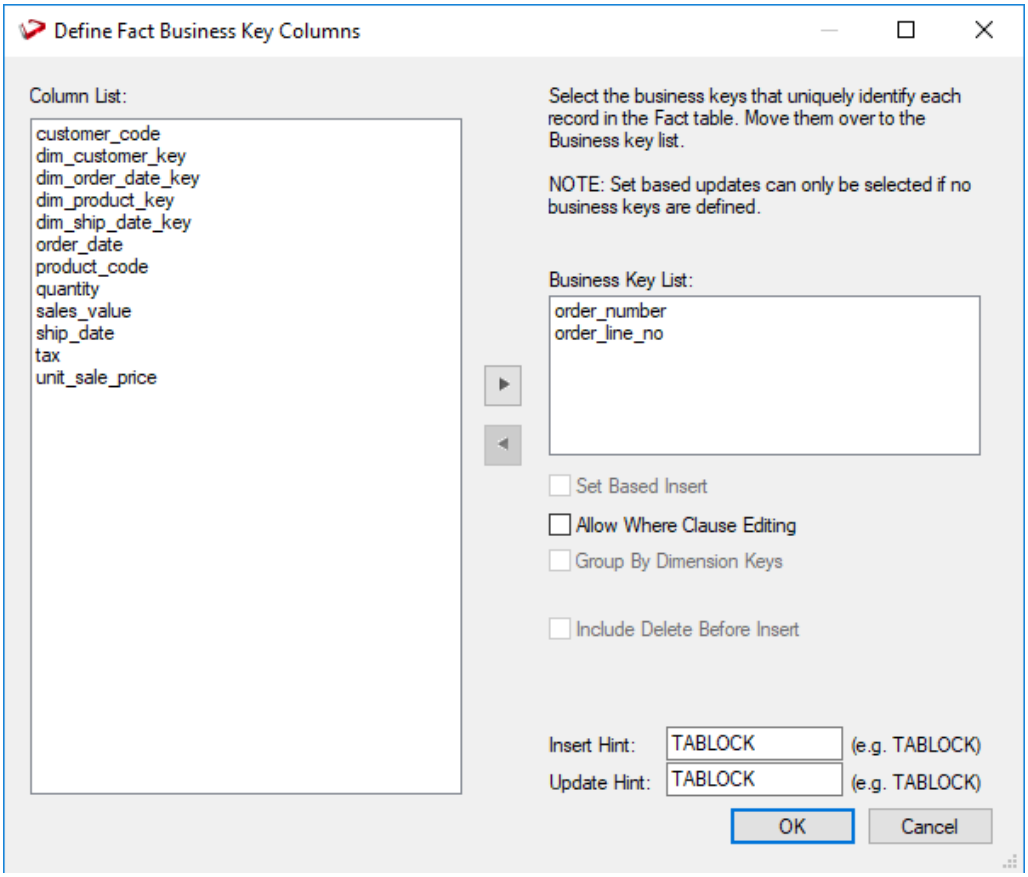
- You can also generate an update procedure via a template, refer to **Rebuilding Update Procedures** for details.
- If the fact table is created in a custom database target, then an additional processing option is available—PowerShell script-based processing. Refer to **Script Templates for Custom Database Table Objects** for details.

Generating a Procedure

1. To generate a procedure, right-click on the fact table in the left pane and select **Properties**.
2. Click on the **Rebuild** button to start the process of generating the new procedure.
3. A series of prompts are displayed during the procedure generation based on the type of load information.

Define Fact Business Key Columns

The first dialog displayed when generating a detail fact table update procedure is the define fact business key columns dialog, asking for the business key that will uniquely identify each fact table record.
The source table from which the fact table is derived would normally have some form of unique constraint applied.
In most cases this will be the business key. In the example below, the order_id and order_line_no are selected for the business key list.



© Copyright, WhereScape Software Ltd All rights reserved.

A business key can be made up of multiple columns, but it must provide a unique identifier. Where multiple columns separately uniquely identify rows in the fact table, choose one to act as the primary business key.

For example a source table may have a unique constraint on both a product code and a product description. Therefore the description as well as the code must be unique.

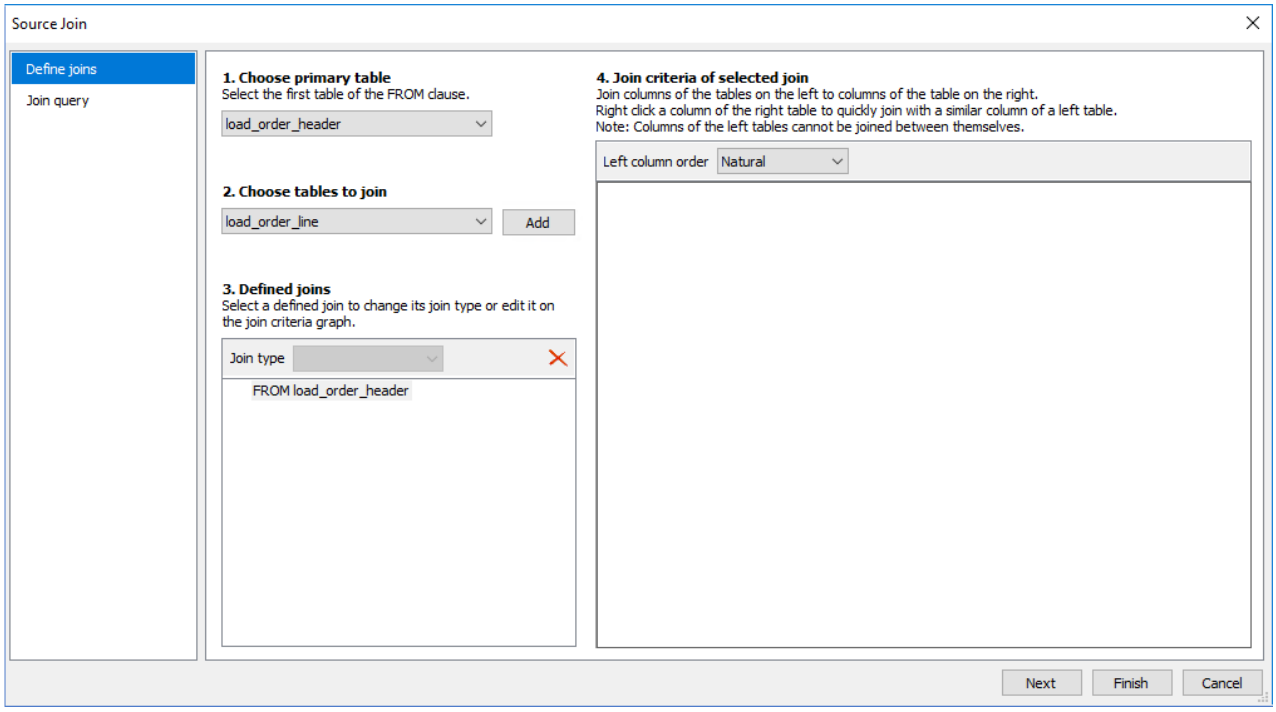
Note

None of the columns chosen as the business key should ever contain a NULL value.

Fields	Description
Allow Where Clause Editing	If the **Allow Where Clause Editing** option is selected, then the next dialog to be displayed is the **Source Join** wizard, else this is skipped.
Insert Hint	Enter a database hint to be used in the INSERT statement.
Update Hint	Enter a database hint to be used in the UPDATE statement. **Oracle Only:** If the **Set Based Insert** and **Include Delete Before** insert options are selected, the **Definition for a Delete Statement before the Fact Table Insert** will appear allowing for options such as **Truncate and Multiple Pass Delete**.

Source Join Wizard

This wizard is used to join source tables and add 'Where' clauses.



Source table joins should have been performed in the stage table, refer to **Generating the Staging Update Procedure** for details.

Since most of the real work is done in the stage table, this is all that is needed to build the Detail Fact Update Procedure.

Rollup or Combined Fact Tables

Rollup fact tables are typically the most heavily used fact tables in a data warehouse implementation. They provide a smaller set of information at a higher level of granularity.
These fact tables typically have a monthly granularity in the time dimension and reflect figures for the month or month-to-date. They often combine multiple analysis areas (detail fact tables) into one super set of information. This allows simpler comparisons or joining of otherwise isolated pools of information.

© Copyright, WhereScape Software Ltd All rights reserved.

An example may be the combination of daily sales, weekly demand forecasts, and monthly budgets, into one rollup fact table. Such a fact table provides the ability to compare monthly or month-to-date sales against budget and forecast.

These rollup/combined fact tables may and often do contain different levels of granularity, as well as different sets of dimensions.

For example, it may be possible to look at sales and forecasts by promotion, but the budget information does not have a populated promotion dimension, so we will not get a budgeted value for the promotion.

Creating Rollup Fact Tables

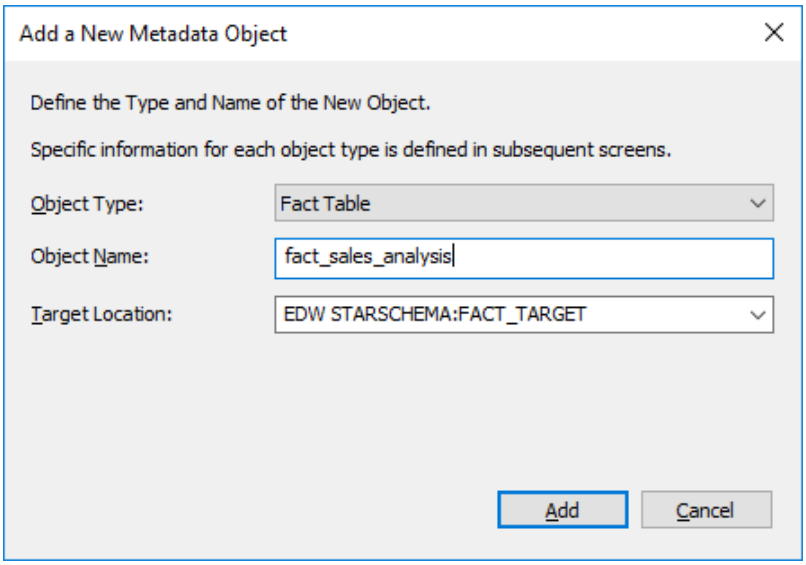
A rollup or combined Fact table is typically created by dragging a Fact table onto a Fact table list. In the example below, a double-click on the Fact table object group under the Sales project produced a list in the middle pane, showing the existing Fact tables.

A **fact_sales_analysis** rollup Fact table is created after selecting and dragging the **fact_sales_detail** table in the right pane into the middle pane.

The initial object create window appears. Note that the name implies that this is a rollup Fact table.

The name is changed to the one chosen above.

A list of columns is displayed and those not required are deleted (everything except additive measures and dimensions).



Drag and drop additional columns from other Fact tables into the middle pane if required. Columns that have the same name need to be changed.

For example, if we acquired a quantity column from **fact_sales_detail** and a column of the same name from **fact_forecast** we may choose to change the names to **actual_quantity** and **forecast_quantity**.

Once all columns have been defined:

1. Create the table.
2. Edit the properties of the table and request that a new procedure be created.
3. Click **OK** and the window to create a template procedure is commenced. This process can include the definition of the date dimension granularity. Monthly rollup is the norm.
4. To just combine Fact tables with no rollup, select the date dimension key as directed in the window. Other forms of dimension rollup require some alteration to the produced template procedure.

Rollup Fact Columns

The columns for a rollup Fact table are typically the dimensions, and only those key measures that are added.

For example, in the sales world, measures may simply be quantity, kilograms, $value, forecast_quantity, forecast_$value, budget_quantity, budget_$value.

© Copyright, WhereScape Software Ltd All rights reserved.

Generating the Rollup Fact Update Procedure

Once a rollup fact table has been defined in the metadata and created in the database, an update procedure can be generated to handle the update of the fact table.

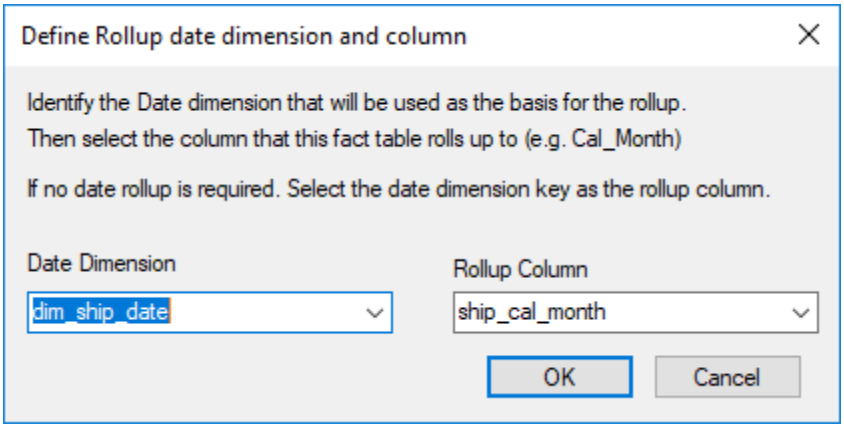
Note: You can also generate an update procedure via a template, refer to **Rebuilding Update Procedures** for details.

Generating a Procedure

1. To generate a procedure, right-click on the fact table in the left pane and select **Properties** to edit the properties for the fact table.
2. Click the **Rebuild** button to start the process of generating the new procedure.
3. A series of prompts are displayed during the procedure generation based on the type of load information.

Define Rollup date dimension and column

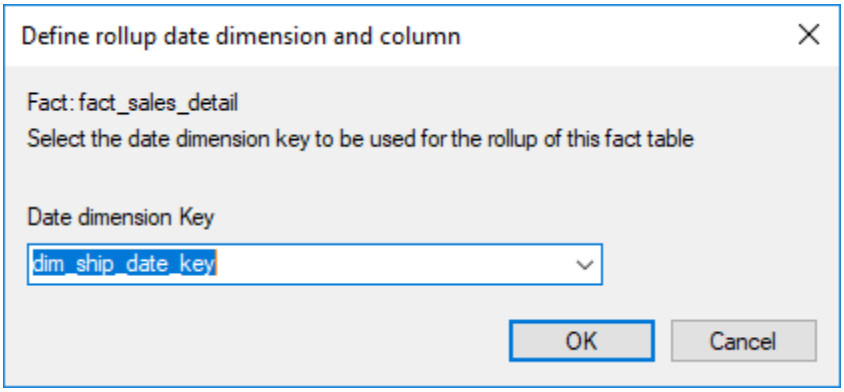
The first dialog displayed when generating a detail fact table update procedure is the define rollup date dimension and column window:



RED needs to know which date is driving the rollup and to which level of detail we wish to set as our base.

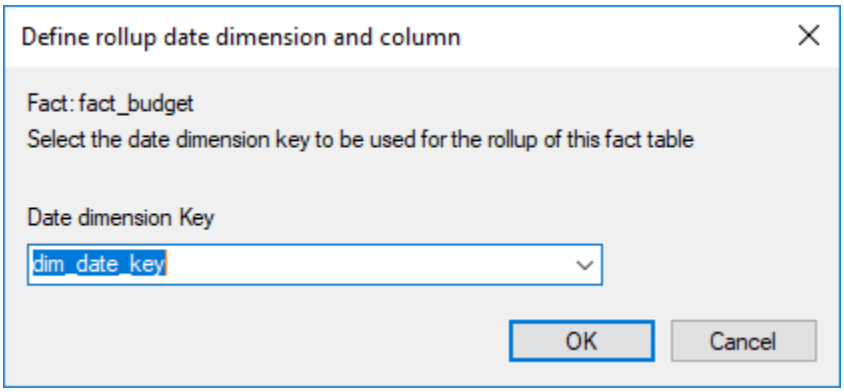
In this example, we have chosen **dim_ship_date** as our date dimension and **ship_cal_month** as our rollup column to set the level to summarize all transactions. Click **OK**.

RED also needs to understand which date field in the actuals should be used to base the rollup; we have chosen **dim_ship_date_key** and click **OK**.



Once again, select the date dimension key and click **OK**.

© Copyright, WhereScape Software Ltd All rights reserved.



KPI Fact Tables

Key Performance Indicator (KPI) fact tables share a lot of similarities with the rollup fact tables in that they are typically at a monthly level of granularity and they combine information from multiple detail fact tables. They differ in that they provide specific sets of pre-defined information (KPIs). Such KPIs are often difficult to answer and require the implementation of some form of business rule. Also they often provide pre-calculated **year-to-date**, **same month last year**, and **previous year-to-date** values for easier comparisons. They have a KPI dimension which provides a list of all the specific statistics or performance indicators we are tracking.

An example from our previous sales examples may be a set of KPIs dealing with customer acquisition and loss. We may have a specific KPI that records the number of new customers during the period and the $value of their business during that period. Another KPI may record the percentage growth of all customers over the last six months. Another may show the number of customers who closed their accounts in the period, and the value of their business over the preceding 12 months.

KPI fact tables require the specific definition of each KPI.

Creating the KPI Dimension

The KPI dimension essentially provides a record of each KPI or statistic we are tracking. There should be one row in this table for each such element.

The KPI fact table requires that this dimension contain at the least an artificial dimension key and a business key. We will refer to the business key as the KPI Identifier.

The following example covers the manual creation of a basic KPI dimension:

1. Right-click the **Dimension** object group in the left pane and select **New Object**.
2. Give the new dimension a name for example **dim_kpi**.
3. Select a **Normal** dimension and click **OK** on the Properties screen to define the dimension table.
4. Right-click the dimension name in the left pane and select **Add Column**, add the three columns defined in the following table.

Field	Column 1	Column 2	Column 3
Column Name	dim_kpi_key	kpi_identifier	kpi_name
Business display	kpi artificial key	kpi unique id	kpi name
Datatype • **Oracle** • **SQL Server/DB2**	integer integer	varchar2(12) varchar(12)	varchar2(256) varchar(256)
Nulls	Clear	Clear	Checked
Key Type	0	A	Clear
Artificial Key	Checked	Clear	Clear

© Copyright, WhereScape Software Ltd All rights reserved.

Field	Column 1	Column 2	Column 3
Primary Business	Clear	Checked	Clear

The first two columns are required, and should be set up as shown above, although a different data type can be used for the second column, and the names can be chosen to suit the requirements.

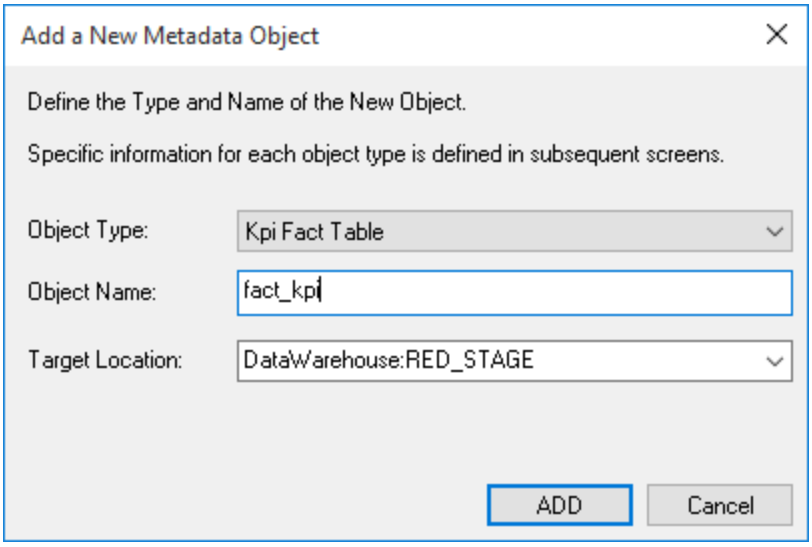
Creating KPI Fact Tables

A KPI fact table is normally created manually.

1. Right-click the Fact Table object group in the left pane and select **New Object**.
2. A dialog displays as shown below.

Note

The default object type is **Fact Table**. This must be changed to **KPI Fact Table** and a table name entered.



Adding columns

Once created, the KPI fact table need to have columns added. New columns can be added by dragging columns from the browser window, or by manually adding from the right mouse menu. Dimension keys should normally be dragged to ensure that all the properties are correctly set.

The columns for a KPI fact table are typically the dimensions and generic measures. For example, valid measures could be **kpi_count**, **kpi_quantity** and **kpi_value**. The interpretation of these measures depend on the KPI or group of KPIs being queried. As with the rollup fact table it is normal and desirable to keep the number of measures as small as possible. Attributes should not be included in these fact tables. These measures must allow Null values for the generated update procedure to function correctly.

Notes

- The kpi measure column order is important. For example if we have mtd count, quantity and value in that order then the prev month, prev ytd and ytd figures also need count first followed by quantity and finally value. If this order is not adhered to then the generated procedure will load the wrong values into the wrong columns.
- A kpi fact table should also have the column dss_update_time added. This column is required if the kpi table is to be used to populate higher level fact tables, aggregates or cubes. This column should be populated with the current date time whenever the particular row is updated.

© Copyright, WhereScape Software Ltd All rights reserved.

Setup of KPI Fact Tables

The set-up of a KPI fact table is somewhat more complex than the other types of fact table. Once all columns have been defined, the following steps need to be taken:

1. Create and populate a KPI dimension that provides at least an id (unique identifier) for each KPI and a description. The key from this dimension must be one of the columns in the fact table.
2. Physically create the KPI fact table in the database once all columns including the KPI dimension key have been added.
3. Define the column type and default value for ALL columns in the table. There must be at least one date dimension key and one KPI dimension key. These attributes are set by selecting a column and modifying its properties.
4. Create the update procedure for the fact table.
5. Set-up each individual KPI.

KPI Setup

KPI Dimension

The KPI dimension provides a means of selecting a specific KPI. This dimension needs to be built in advance of the KPI fact table. It does not need to be populated, as the KPI fact table will add entries to this table as required. A key and some form of unique identifier for the KPI is required, e.g. they may be numbered or given short names.

Create the KPI Fact Table

Right-click the KPI table and select **Create/Recreate** to physically create the table in the database.

Defining column types and default values

1. List all the columns for the KPI fact table.
2. Amend the properties for each column. The column types can be one of the following:

Type	Description
Dimension Key	Any dimension that is not the time or KPI dimension
Time Dimension Key	Time dimension key
KPI Dimension Key	KPI dimension key
Month-to-date Measure	Used to identify the normal KPI measures. These are calculated to populate the KPI during each run.
Year-to-date Measure	Used to identify year-to-date measures for the KPI. Optionally calculated or derived from existing rows and measures in the KPI fact table.
Last year same month Measure	Used to identify measures for same month last year comparisons. Derived from existing rows and measures in the KPI fact table.
Previous year-to-date Measure	Used to identify previous year-to-date measures. Derived from existing rows and measures in the KPI fact table.
Fact Table Key	Not normally used
Date	Not normally used
Attribute	Not normally used
Measure	Not normally used

Tip

If an **Attribute** or **Measure** column is specified then this column will just be set to its default value. In this way an update_time column can be set by making its default value **sysdate**.

© Copyright, WhereScape Software Ltd All rights reserved.

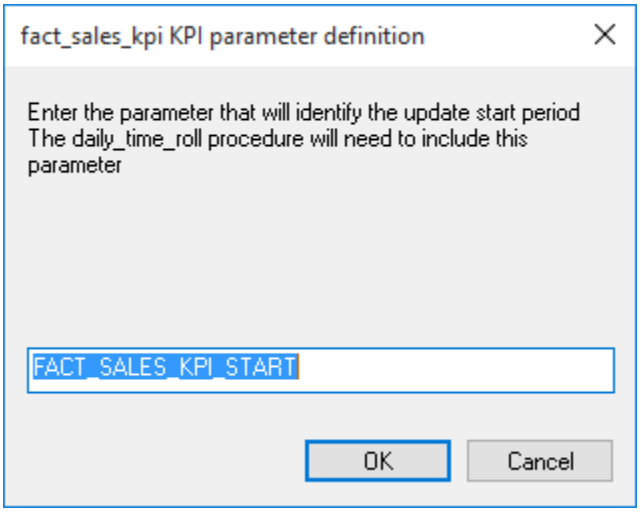
For each column set a default value.

- For measures and dimension keys this would normally be zero (0). Nulls can be represented as two single quotation marks, i.e. ' '
- **sysdate** can be used for a default date value
- These default values populate the columns where and when no other information is available. For example if a KPI only makes use of 4 of the 6 dimensions available then the remaining 2 dimensions will be set to their default values.

Create the 'Update' procedure

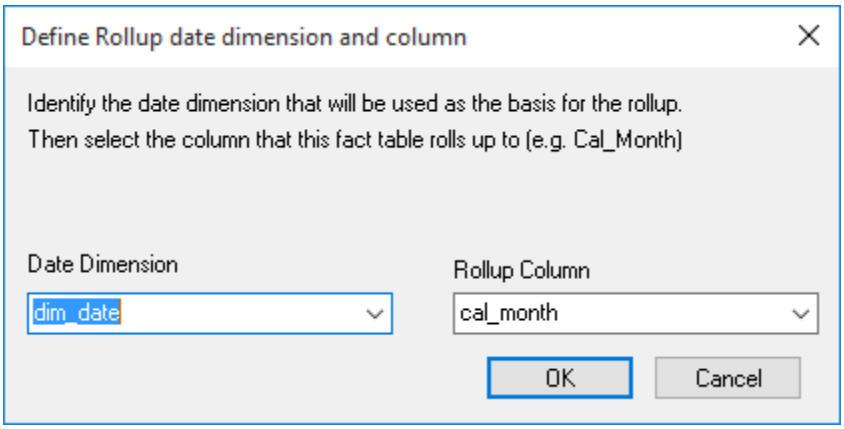
Right-click the KPI fact table name and select **Properties**. In the Properties window, set the update procedure list to the **(Build Procedure...)** option. Click **OK** to initiate a series of window prompts to create a procedure used to update the kpi fact table.

The first two dialogs ask for parameter names to define the start and end periods for which the KPIs will be updated. The default naming convention is fact_table_name_START and fact_table_name_END. Accept these default names if possible. These two parameters need to be created, and need to be updated once the KPIs have been defined and tested. Under normal operation these parameters should probably be set daily from the daily_time_roll procedure.



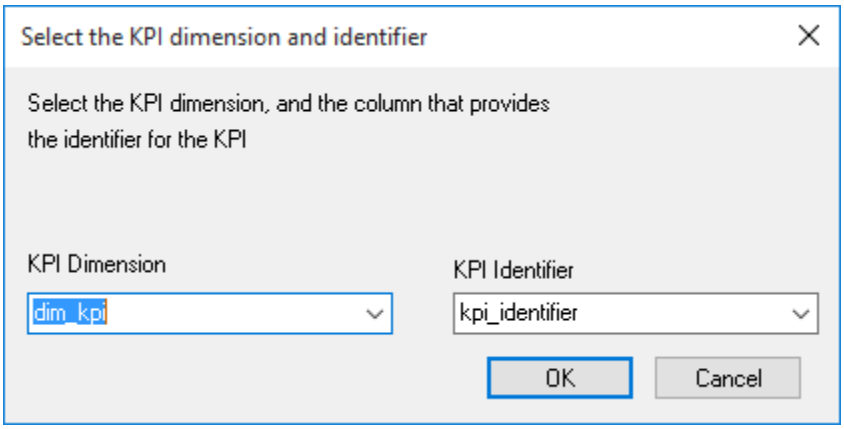
The date dimension and column for rollup is required. Normally this would be cal_month or fin_month.

This is based on the assumption that most KPIs are monthly or annual. Subsequently a rollup year and date will be required from the same date dimension.



The KPI dimension is required. Also the identifier column (not the key) that uniquely identifies the KPI. This would normally be a number or short name (max 64 chars).

© Copyright, WhereScape Software Ltd All rights reserved.



The procedure is then created and compiled.

Define each KPI

Once the procedure has successfully compiled, we are ready to start setting up each of the KPIs. Right-click the KPI table name to get the pop-up menu.

This menu is similar to the other fact table menus except for the additional options to display and add KPIs. A display of the KPIs will list any KPIs that are currently defined. The result set is as appears below:

KPI List for fact_sales_kpi

KPI	Name	Active	Method	Description
Activity	Customer activit...	Enabled	Statement	Provides a count of active custom
Activity	Customer activit...	Enabled	Statement	Provides a count of active custom
Activity	Customer activit...	Enabled	Statement	Provides a count of active custom
Sales	Customer sales (...	Enabled	Statement	Provides a count and dollar value
Sales	Customer sales (...	Enabled	Statement	Provides a count and dollar value
Sales	Customer sales (...	Enabled	Statement	Provides a count and dollar value

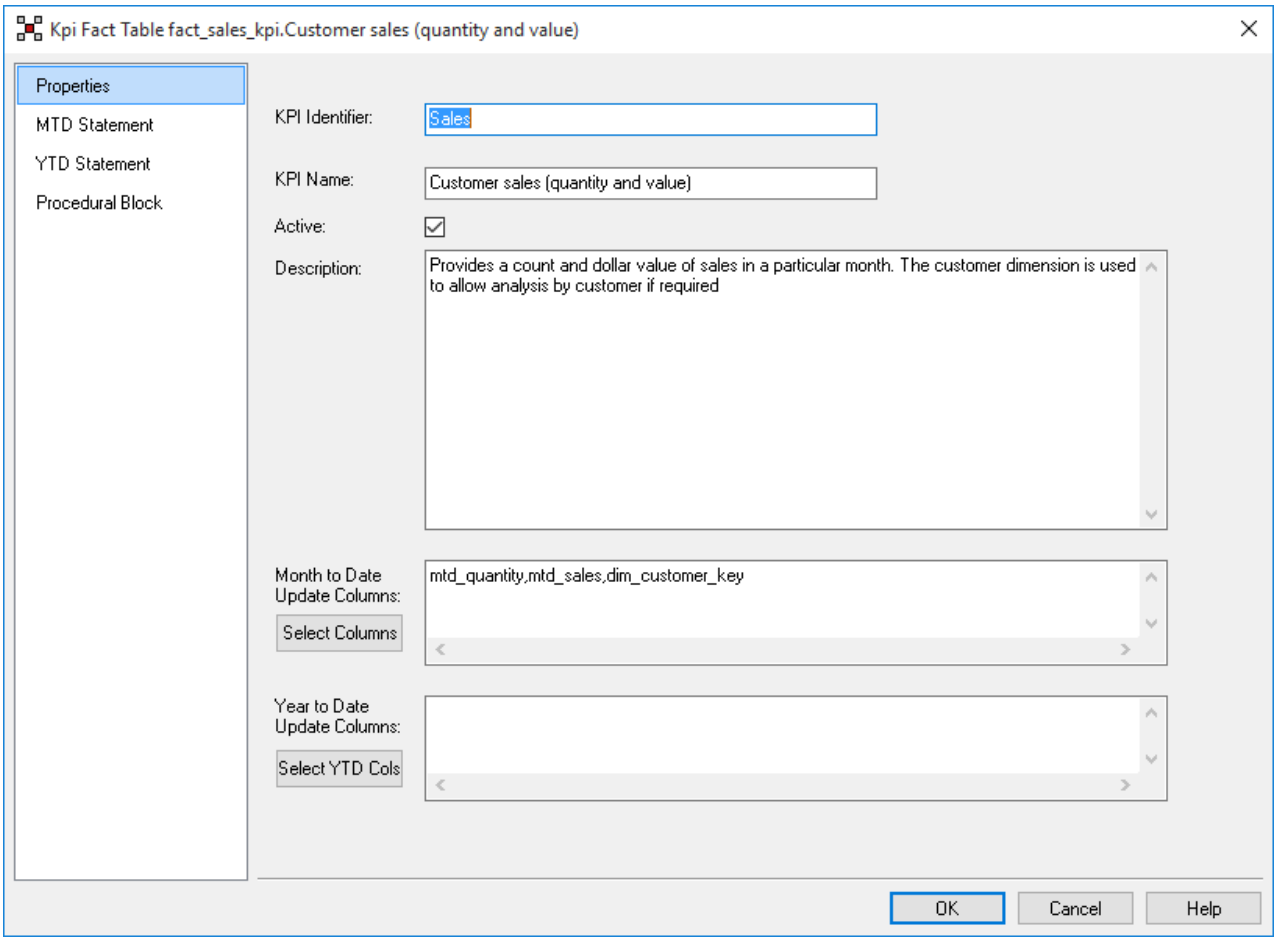
The 'Id' is the same unique identifier that will be found in the kpi dimension. The **Name** is a name entered in the properties of the KPI and does not relate to the dimension in any way. The **Active** option indicates if the KPI will be processed when the **Update** or **Process** menu options are selected. Each KPI can be defined as a SQL Statement, a Procedural Block or a Procedure. The current method used to update the KPI is one of these three and is displayed under **Method**. The description is a comment or Description field entered in the Properties of the KPI and in no way relates in the KPI dimension.

Right-click anywhere in the List window to access the KPI add option. Clicking the right mouse button while positioned over the 'Id' opens a pop-up menu which provide options to copy, modify, delete, add and view the KPI statement.

When the add KPI option is selected, a Properties screen appears as shown below. The id and name fields are required. If the **Active** check box is selected ,the procedure is enabled for update.

The **Select Columns** button enables the selection of the columns that this KPI will update during the month to date update pass. The date and KPI dimensions will be updated for you so are not provided as choices. The only choices provided are the 'Month to date measures' and the other 'Dimensions'. Select those measures and dimensions that will be updated. All non-selected measures and dimensions will be set to their default values. The resultant choice is shown in the **Update Columns** window. This window should only be updated via the **Select Columns** button.

© Copyright, WhereScape Software Ltd All rights reserved.



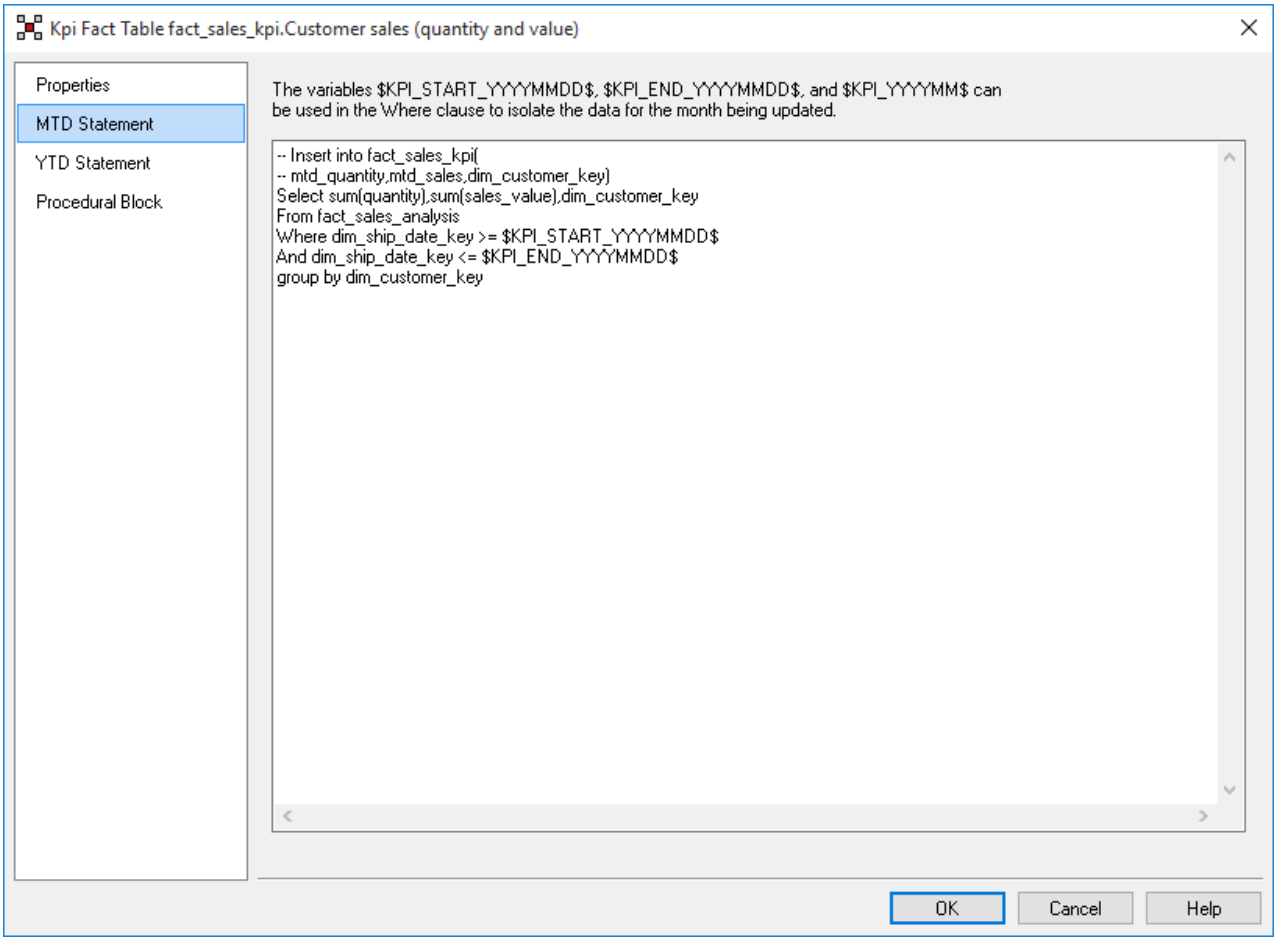
Once these initial values are defined, the **Statement** tabs can be selected to define the update SQL statement. If a procedural block or procedure is required for the update (i.e. too complex for a single SQL statement), then select the **Procedural Block** tab.

© Copyright, WhereScape Software Ltd All rights reserved.

The KPI Statement

The **Statement** tabs present edit windows that enable the input of a SELECT statement that will return the values for the columns being updated.

They must return the values for the columns selected in the **Properties** tab in the order shown in the **Update Columns** window. An example of a completed KPI statement follows:



The first two lines are comments inserted (when modifying) to provide a reminder of the columns and order required by the select statement. Anything prior to the initial SELECT statement is ignored.

The statement must make use of at least one of the variables provided to select the appropriate period. This variable is replaced with the values for the period being updated at run time. These values are derived from the start and end parameters defined for the KPI fact table.

In order to group by the non-selected columns and any attributes, etc. WhereScape RED must be able to break this statement into its component Select, From, Where, Group By, and Having components (where present). If a complex statement is used that has more than one of these key words then the primary one must be in UPPER case. For example if a sub query is being used then the main SELECT and FROM key words must be in upper case and must be the only occurrence of these key words in uppercase. WhereScape RED checks and issues a warning if it is unable to resolve the main key words.

To test a statement, right-click the KPI Id and select **View Statement**. This provides a window with the full statement with the default columns inserted and the parameters converted. This statement can be cut and run in SQL Admin (shipped with WhereScape RED) or in a database specific utility for testing.

Procedural Block

The **Procedural Block** tab presents an edit window that enables the input of an Oracle anonymous block. If there is anything in this window it is actioned, instead of the statement. A button in the lower right allows a rudimentary template block to be built. If used, this block must do the actual insert statement rather than just the select. It must also handle ALL dimension, month to date measure and non standard columns. In addition, it will need to put default values in columns not actively being updated. The KPI dimension and date dimension keys must also be updated. The output from a 'View Statement' option for another KPI gives an indication of

© Copyright, WhereScape Software Ltd All rights reserved.

what the block must do. The same month last year, year to date and previous year to date measures are still handled automatically. The same $KPI_ variables as shown above the statement edit window can and must be used.

Procedure

If for some reason it is not possible to implement the KPI in either the statement or procedural block then a procedure may be created. The name of such a procedure can be entered in the window at the bottom of the **Procedural Block** tab. If any value is present in this window, then the procedure takes precedence over both the statement and block.

This procedure must adhere to the standard set of parameters as with all other procedures to be executed by the scheduler. It must undertake the same tasks as defined above in the **Procedural Block** section, and as with those tasks it does not need to set the year to date, last year same month and previous year to date values.

Copying KPIs

Right-click the KPI Id and select **Insert Copy** to insert an exact duplicate of the KPI. This copy can then have its Id and Name changed along with its statement to create a new KPI. This is particularly useful when only small differences exist between a number of KPIs.

Testing/Development

During testing it would be normal to DISABLE all KPIs except the one currently being developed. In this way the update or process options will only utilize that KPI. As mentioned above the ‘View Statement’ option of the KPI popup menu displays, and allows the cutting of, the statement that will be executed to acquire the KPI information. This statement can be tested in SQL*Plus or some other environment if problems are encountered.

© Copyright, WhereScape Software Ltd All rights reserved.

Generating the KPI Fact Update Procedure

Once a Detail Fact table has been defined in the metadata and created in the database, an update procedure can be generated to handle the update of the Fact table.

Note

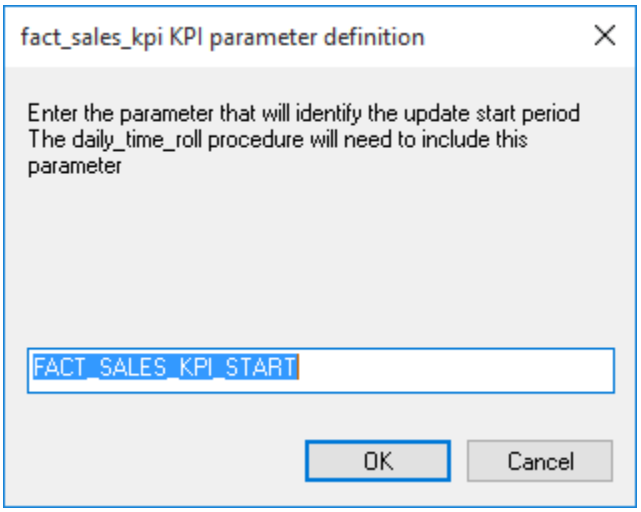
You can also generate an update procedure via a template, refer to **Rebuilding Update Procedures** for details.

Generating a Procedure

1. To generate a procedure, right-click the Fact table in the left pane and select **Properties**.
2. Click the **Rebuild** button to start the process of generating the new procedure.
3. A series of prompts are displayed during the procedure generation based on the type of load information.

KPI parameter definition

The first window displayed when generating a Detail Fact table update procedure is the KPI parameter definition dialog:



The **KPI parameter definition** window prompts for a parameter to hold the start period that we are interested in. This period sets the first period that is processed when the KPI Fact table is updated. Once entered, click **OK**.

Note

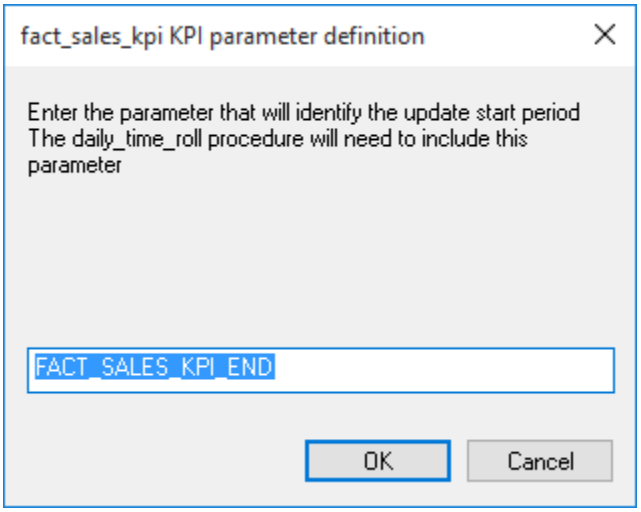
If this parameter is not set, the default current period is set from dim_date. The format is YYYYMM.

Tip

The **dim_date** custom update procedure **daily_date_roll** can be updated to set these parameters automatically if you do not wish to control them manually. This is good practice.

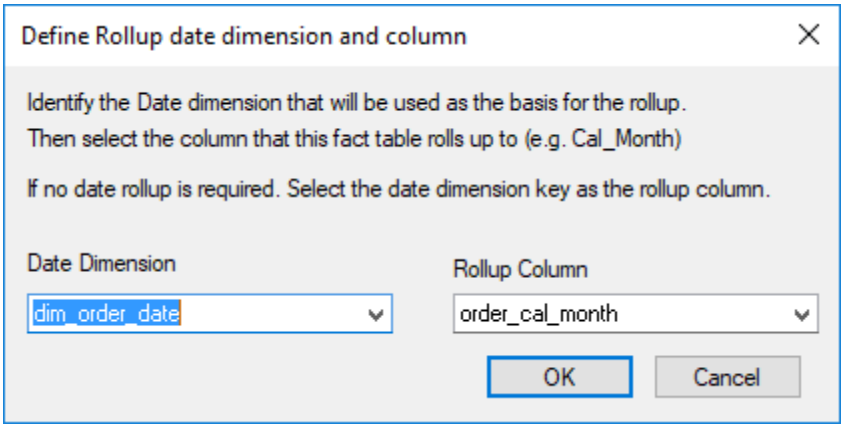
© Copyright, WhereScape Software Ltd All rights reserved.

Similarly, a **KPI parameter definition** dialog prompts for a parameter to hold the end period. Once entered, click **OK**.

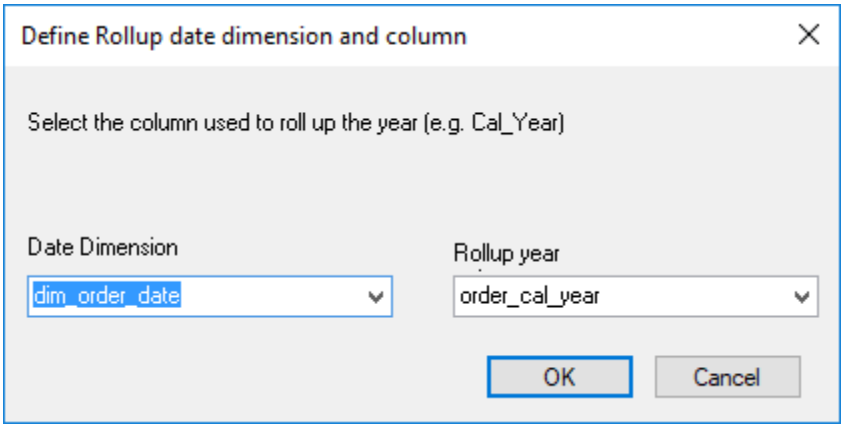


Define Rollup date dimension and column

As with Rollup tables, we now need to specify which date in the transactional data we will use to control the rollups and to which level – in this example, for the Monthly level rollup, we choose **dim_order_date** as our **Date Dimension** and **order_cal_month** as our **Rollup Column** and click **OK**.

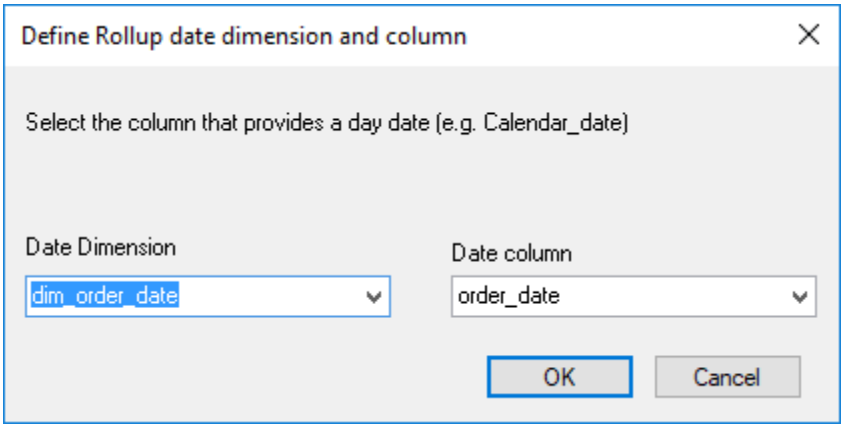


Once again for our Yearly rollup, in this example we choose **dim_order_date** as our **Date Dimension** and **order_cal_year** as our **Rollup Year** and click **OK**.



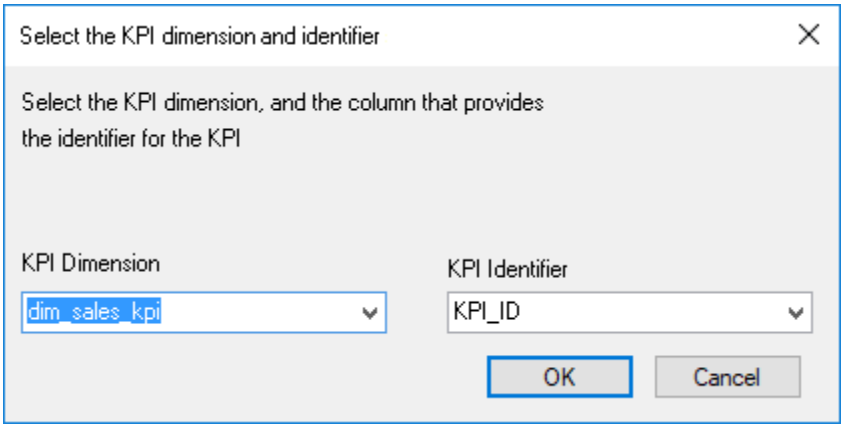
© Copyright, WhereScape Software Ltd All rights reserved.

The next screen sets the Daily level for the update procedure to understand the lower level detail, we choose **dim_order_date** as our **Date Dimension** and **order_date** as our **Date Column** and click **OK**.



Select the KPI dimension and identifier

Finally we need to specify where the KPIs are defined and how they are defined, so in this example we choose **dim_sales_kpi** as the **KPI Dimension** and **KPI_ID** as the **KPI Identifier**. Click **OK.**



© Copyright, WhereScape Software Ltd All rights reserved.

Snapshot Fact Tables

Snapshot fact tables provide an 'as at' view of some part of the business. They always have a time dimension that represents the 'as at' date. For example, we may have a weekly or monthly snapshot of our inventory information. Such a snapshot would allow us to see what inventory we had at a particular point in time and allow us to compare our holdings over time. Snapshots are often used when dealing with people profiles or client/customer profiles to enable us to view the state of our relationship with an entity at a given point in time e.g. to see a monthly trend of a client's outstanding balance.

Creating Snapshot Fact Tables

The creation of snapshot fact tables is very dependent on the method used to create the snapshot. In some cases they may be created in exactly the same way as detail fact tables.
These cases are typically where the OLTP application provides the 'as at' state. For example, an inventory system may provide a weekly stock take table which is the basis for our snapshot fact table. In such cases the table is created as per a detail fact table, but with additional tailoring around the time dimension.

In other situations, the snapshot is built from an existing transactional fact table, where all information up to a specific date is summarized or frozen to create the snapshot. In these instances, the snapshot can be created as a rollup fact table.

Regardless of whether a detail or rollup method is used for the creation of the snapshot table, a significant amount of work will be required to tailor the update procedure to reflect the particular requirements of the snapshot table.

Snapshot Fact Columns

The columns for a snapshot Fact table vary to a large degree. A date dimension is always present to reflect the 'as at' date. Some snapshots may have a large number of measures, reflecting $values quantities etc., whereas others may be rich in attribute (non-additive) facts.

Generating the Snapshot Fact Update Procedure

Once a snapshot Fact table has been defined in the metadata and created in the database, an update procedure can be generated to handle the update of the Fact table.

Note

You can also generate an update procedure via a template, refer to **Rebuilding Update Procedures** for details.

Generating a Procedure

1. To generate a procedure, right-click the Fact table in the left pane and select **Properties**.
2. Click the **Rebuild** button to start the process of generating the new procedure.
3. A series of questions is asked during the procedure generation based on the type of load information.
4. If a **detail** method was used for the creation of the snapshot table, refer to **Generating the Detail Fact Update Procedure** for details.
5. However, if a **rollup** method was used for the creation of the snapshot table, refer to **Generating the Rollup Fact Update Procedure** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

Slowly Changing Fact Tables

Slowly changing fact tables provide a complete history of changes for some part of the business. Typically, such fact tables are used for situations where a relatively small number of changes occur over time, but we need to track all of these changes.

A good example is an inventory stock holding that does not change very often. In this case, a slowly changing fact table can record both the current inventory levels as well as providing a means of seeing the state of inventory holdings at any point in time.

Creating Slowly Changing Fact Tables

WhereScape RED does not provide any automated way for creating a slowly changing fact table. The best method is to proceed as follows.

1. Drag and drop the Stage table into a dimension target and create a dummy slowly changing dimension with the name we will be using for our fact table. (i.e. fact_...).
2. Answer the questions to the pop-up dialog boxes. Select a join if asked, but do not bother to join any of the tables. Select the columns to be managed as slowly changing. Normally these will be the measures such as quantity and all of the dimension keys. The generated procedure will fail to compile.
3. Inspect the additional columns added for the dimension and make a note of them. Specifically dss_end_date, dss_start_date, dss_current_flag and dss_version.
4. Inspect the indexes created to help manage the slowly changing dimension and make a note of their columns.
5. Delete the dummy dimension you have created.
6. Delete the get_..._key procedure created for the dimension. Do not delete the update procedure.
7. Create a detail fact table in the normal manner.
8. Add in the special columns as created for the dimension. Namely dss_end_date, dss_start_date, dss_current_flag and dss_version.
9. Recreate the table
10. Create indexes as per the dimension table to help maintain the slowly changing fact table.
11. Modify the update procedure for the fact table. Bring up the old version of the procedure that was created for the dimension of the same name. This would have been automatically versioned out when the latest procedure was created and can be seen through the procedure viewer.
12. This new procedure needs to merge the code from the old procedure, which will provide a guide in how to build the code for a slowly changing fact table.

Partitioned Fact Tables

Partitioned fact tables are normally used for performance reasons. They enables you to break a large fact table into a number of smaller tables (partitions).
WhereScape RED supports partitioned fact tables. The following section explains their creation and support.

Creating a Partitioned Fact Table in SQL Server

In WhereScape RED, a partitioned fact table can only be created from an existing fact table. The process therefore is to create a fact table normally, including the update procedure and indexes.

The table is then partitioned through the **Storage** screen of the fact table's Properties window. WhereScape RED assists in creating the exchange table, modifying the indexes and building the procedure to handle the update of the partitions.

Create a detail or rollup/combined fact table

- Create the fact table that you wish to partition in the usual way.
- You must create an update procedure but do not need to load any data. The process of creating the update procedure will also build any indexes for the fact table.
- RED will be using a partition exchange table to move data to and from the partitioned fact table. This exchange table must have the same columns, column order and indexes as the fact table. It is therefore recommended to get the fact table into its final form before partitioning it.
- Add any indexes to avoid extra work later.

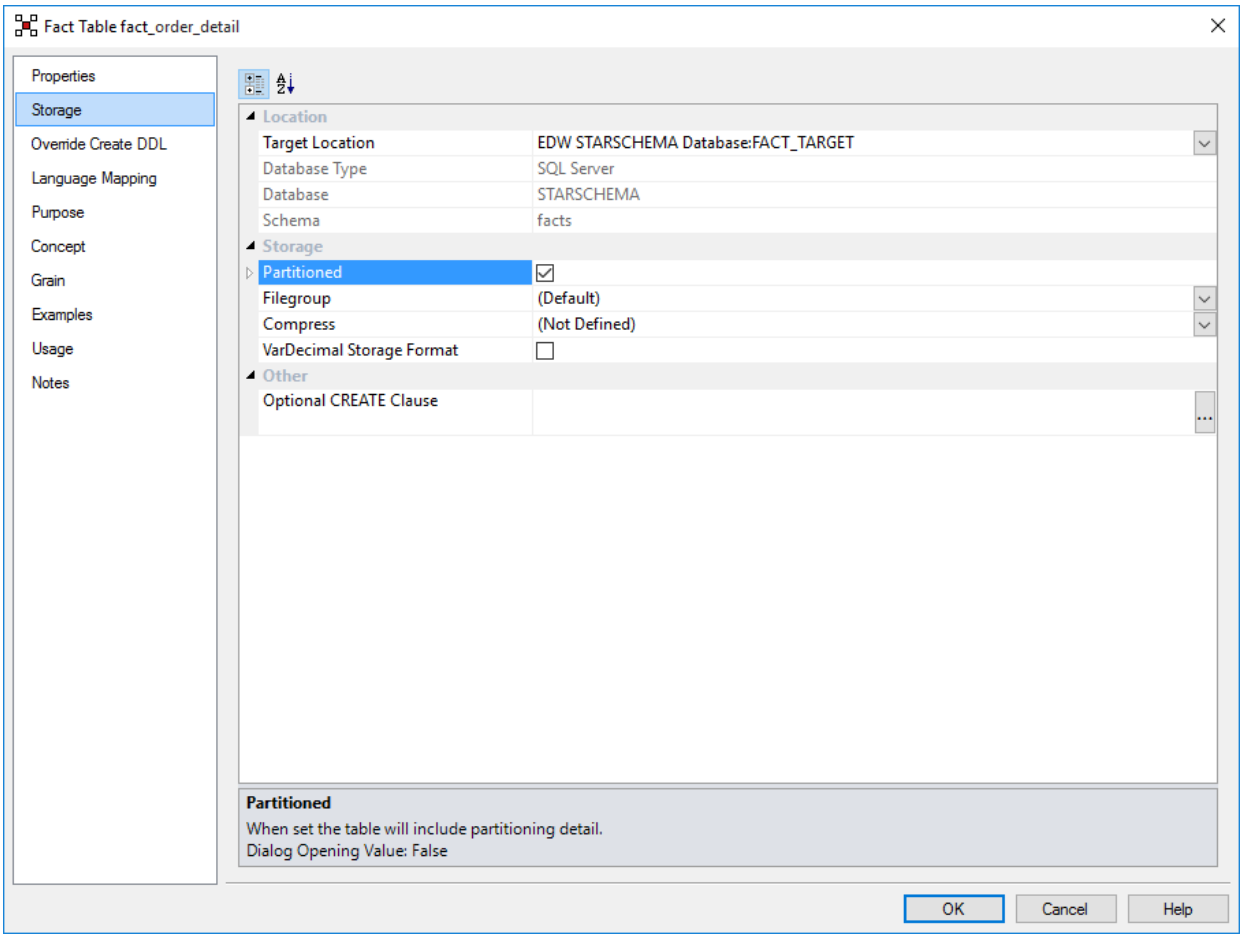
© Copyright, WhereScape Software Ltd All rights reserved.

Convert to a partitioned fact table

1. Double-click the fact table that you want to partition to bring up its properties screen.
2. Click the **Storage** tab and tick the **Partitioned** check-box (see example in the screens below).
3. A prompt is displayed warning you that this table needs to be recreated to change the current fact to a partitioned table and offering assistance to create the partitioned table. The exchange table is then used to replace the relevant period in the fact table. The fact table remains in a queryable state throughout the process.
4. Click **Yes** to launch the wizard that will assist you through the partitioning of the table.

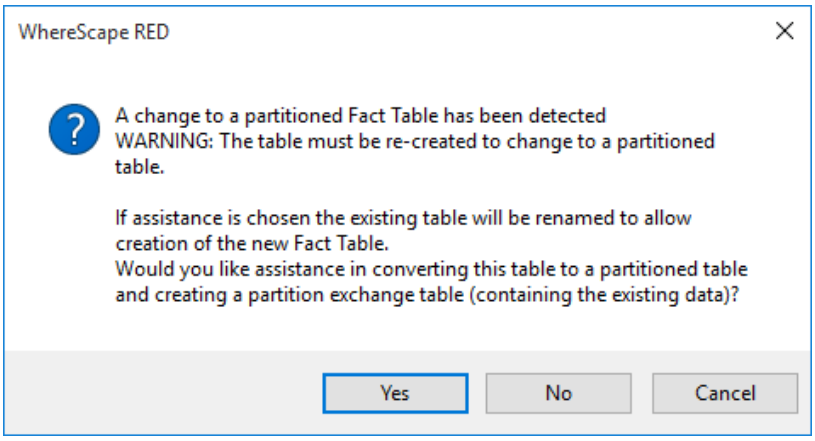
Notes:

Converting an existing non-partitioned table to a partitioned table cannot be done using a deployment application.
To convert non-partitioned to partitioned tables using deployment applications, may require some manual intervention to update the target databases to match the new metadata.

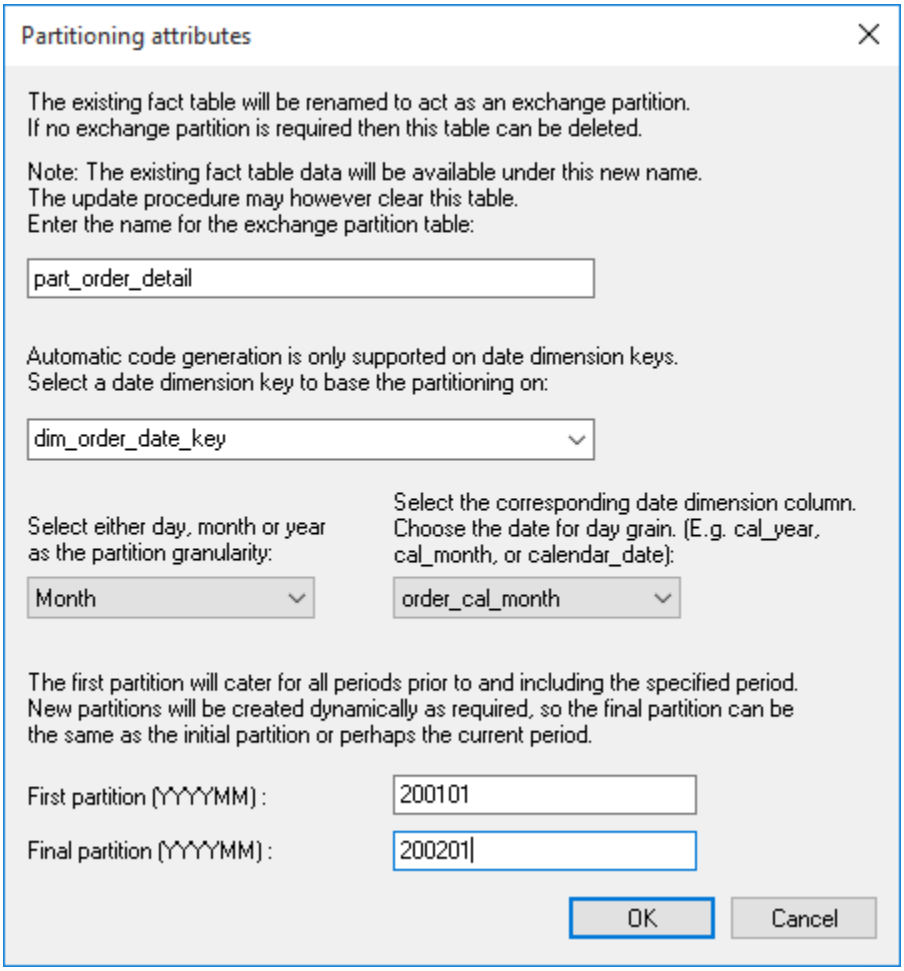


5. As you click the Partitioned check-box, a dialog pop-up is displayed as follows asking for confirmation that the table is to be converted to a partitioned fact table.
   If you click **Yes**, the existing fact table is renamed to become the exchange partition table. Any current data will remain in this table.

© Copyright, WhereScape Software Ltd All rights reserved.



6. The parameters for the partitioned fact table are then prompted for and are described below:



7. The partition exchange table name is prompted for. This table is identical to the fact table and is used to swap partitions to and from the fact table. Enter a **name** for the exchange partitioned table.
   - The fact table will be partitioned by the date dimension key. Select a **date dimension key** that will be used as the basis for the partitioning of the fact table. It is assumed that this key has the standard WhereScape format of YYYYMMDD.
   - Select a partition granularity of day, month or year along with the corresponding column in the date dimension.
     - For a **daily partition** a column with a date data type must be chosen. The date dimension key will actually be used for the partitioning but the date column is required.
     - For a **monthly partition** a column with the format YYYYMM must be chosen. In the standard WhereScape date dimension this is the column cal_month.
     - For a **yearly partition** a column with the format YYYY must be chosen. In the standard WhereScape date dimension this is the column cal_year.

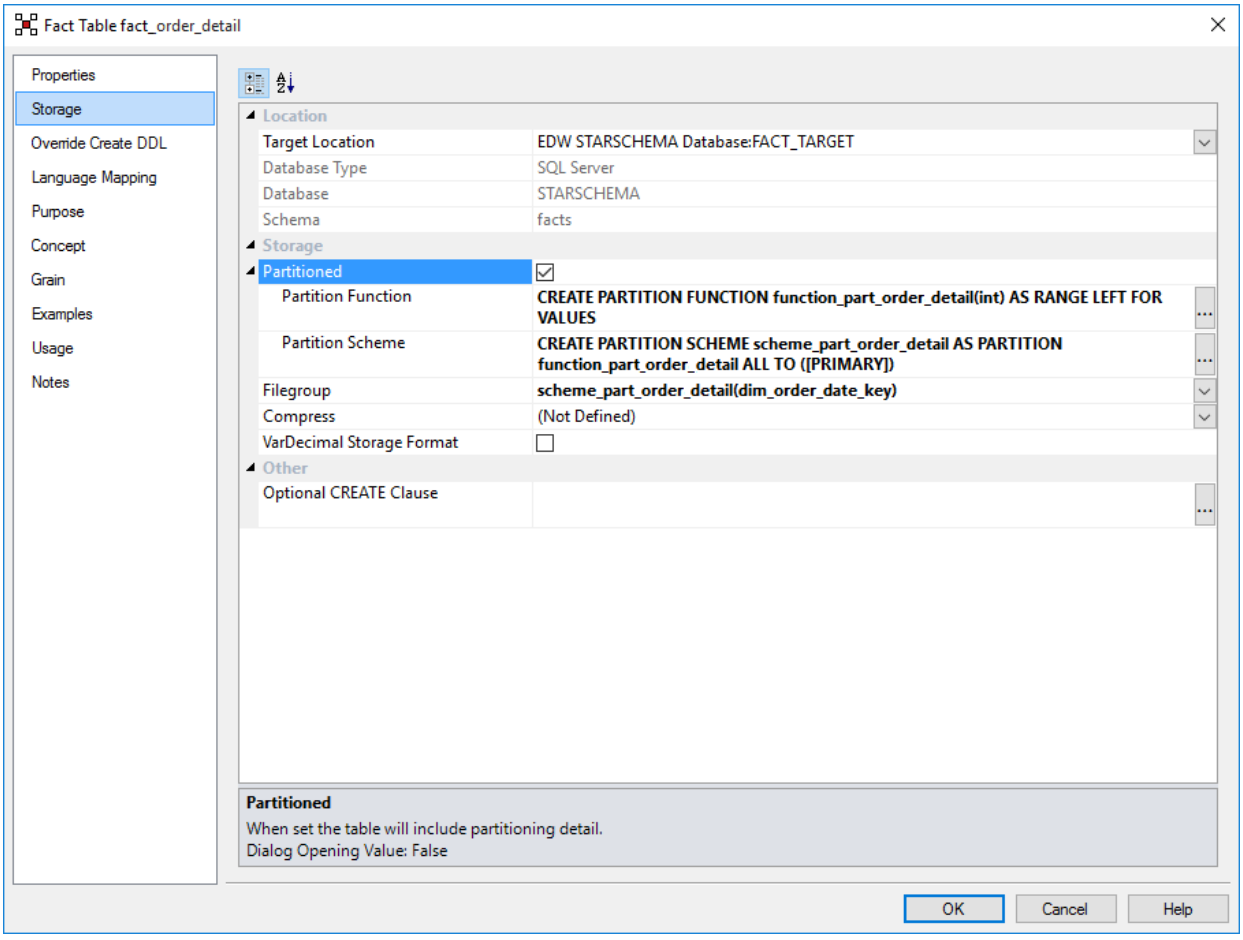
© Copyright, WhereScape Software Ltd All rights reserved.

- The first and last partition should then be selected. The last partition is not that important as additional partitions will be created as required. Normally just select a partition a few on from the first. The first partition is however important as any data that is dated prior to the first partition will be loaded into this first partition. The partition must be entered in the format shown. For example monthly partitions require the format YYYYMM.
- After all the relevant fields have been completed, click **OK** and the conversion process will populate the **Partition Function, Partition Scheme** and **Table Partition Scheme** fields of the table's storage properties.

Note

This field is limited in size, so if too many partitions are chosen only the first few will be added to this field. The other partitions will be created dynamically by the update procedure.

8. Click the **OK** button on the table's **Storage** screen to begin the process of converting the fact table, creating the new fact table and building the new update procedure.
Examine the results screen to see what has been done.



The exchange partition table will now have any current data, to populate the fact table, copy the data from partition table to the fact table, drop and recreate fact index.
The date dimension key used to base the partitioning on is not allowed to have nulls. WhereScape RED has modified the table property, however the partition table needs to be validated and recreated to be valid.

Then the fact table can be processed.

© Copyright, WhereScape Software Ltd All rights reserved.

The partitioned fact table update procedure

The generated procedure will handle the creation of new partitions as required.

Any data for periods prior to the first partition is placed in the first partition. If data is received for a period after the last partition then new partitions are created.

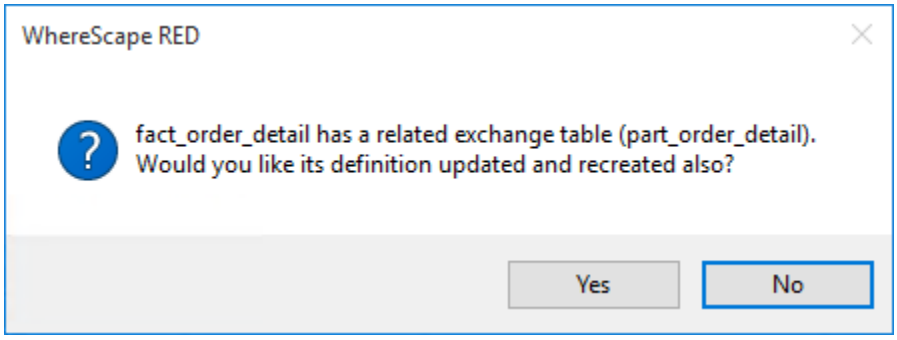
Empty partitions are created if necessary to skip periods where no data is present. For example, if we have a table partitioned by month and the latest partition is 200204 (April/2002). If data is then received for say September 2002 the update procedure will build partitions for May, June, July, August and September.

There is a variable at the start of the update procedure called v_max_skip_periods. This variable is by default set to 12. This defines the maximum number of continuous partitions we will leave empty.

From our previous example if our latest partition was April 2002 and we received data for July 2003 with no interim data then the update procedure will fail as it would have to skip 14 partitions. This check is present to prevent erroneous data from creating hundreds of partitions into the future.

Note

Recreating or Altering a partitioned fact table prompts with an offer to resync and recreate the related exchange table and indexes. This will resync the columns and indexes from the fact table to its exchange partition table so that partition swapping works correctly.



Creating a Partitioned Fact Table in Oracle

In WhereScape RED a partitioned fact table can only be created from an existing fact table. The process therefore is to create a fact table normally, including the update procedure and indexes.
The table is then partitioned through the **Storage** screen of the fact table's properties dialog. WhereScape RED will assist in creating the exchange table, modifying the indexes and building the procedure to handle the update of the partitions.

Create a detail or rollup/combined fact table

- Create the fact table that you wish to partition in the usual way.
- You must create an update procedure but do not need to load any data. The process of creating the update procedure will also build any indexes for the fact table.
- RED will be using a partition exchange table to move data to and from the partitioned fact table. This exchange table must have the same columns, column order and indexes as the fact table. It is therefore recommended to get the fact table into its final form before partitioning it.
- Add any indexes to avoid extra work later.

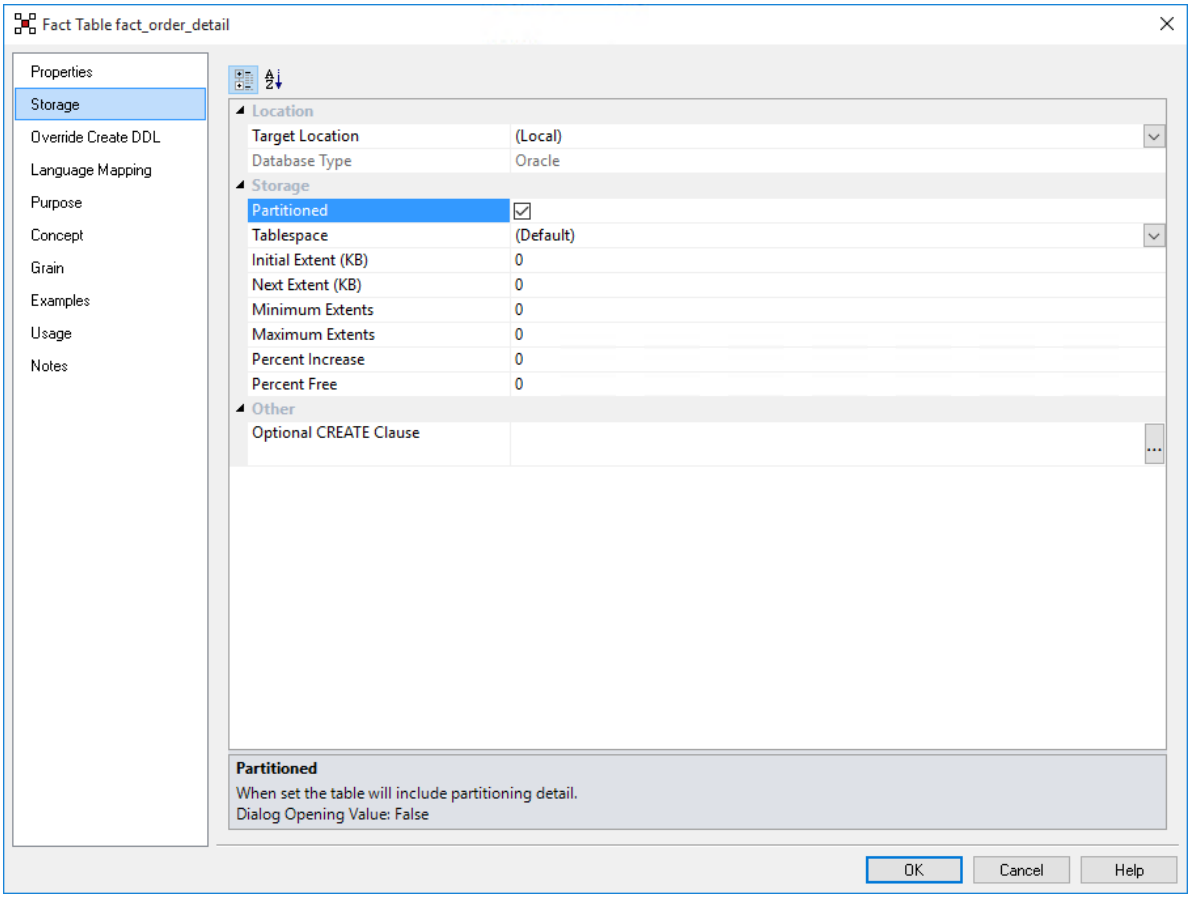
Convert to a partitioned fact table

1. Double-click the fact table that you want to partition to bring up its properties screen.
2. Click the **Storage** tab and tick the **Partitioned** check-box (see example in the screens below).
3. A prompt is displayed warning you that this table needs to be recreated to change the current fact to a partitioned table and offering assistance to create the partitioned table. The exchange table is then used to replace the relevant period in the fact table. The fact table remains in a query able state throughout the process.
4. Click **Yes** to launch the wizard that will assist you through the partitioning of the table.

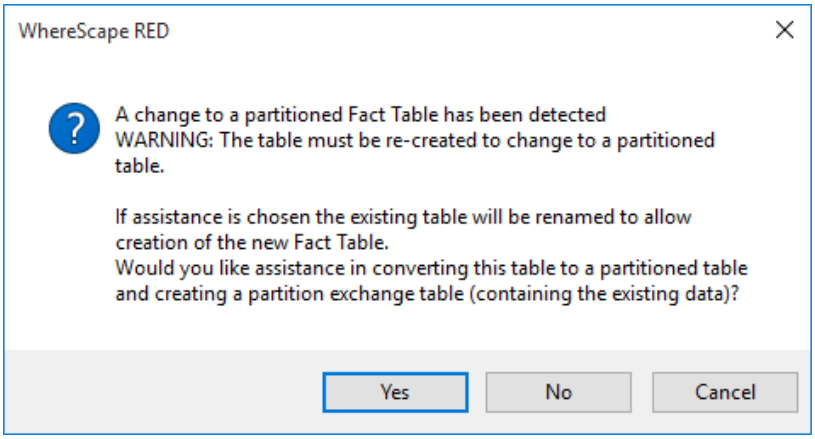
© Copyright, WhereScape Software Ltd All rights reserved.

Notes:

Converting an existing non-partitioned table to a partitioned table cannot be done using a deployment application.
To convert non-partitioned to partitioned tables using deployment applications, may require some manual intervention to update the target databases to match the new metadata.

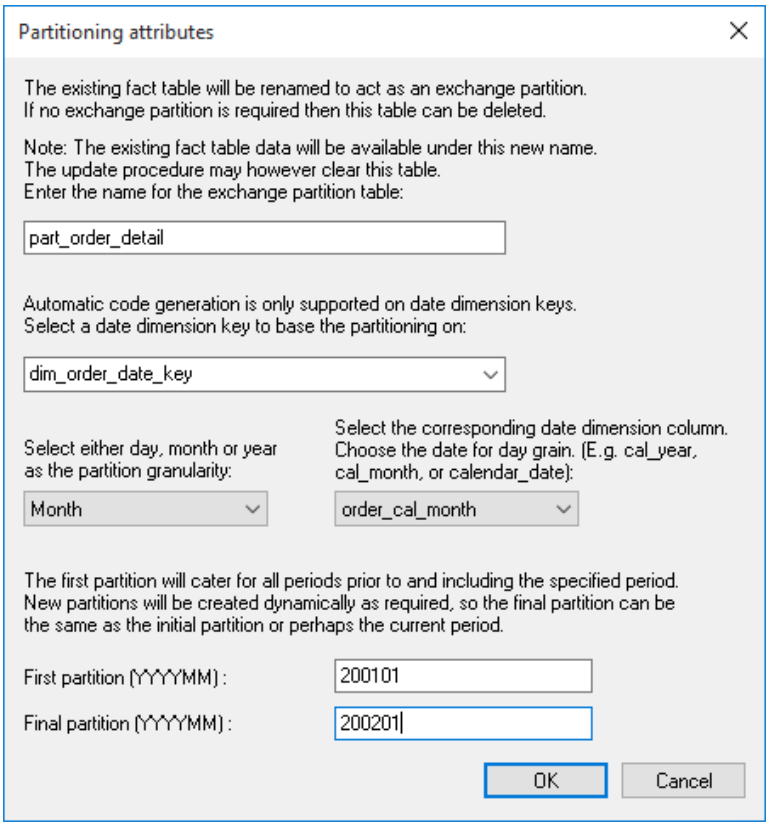


5. A dialog will pop-up as follows, asking for confirmation that the table is to be converted to a partitioned fact table.
   If you click **Yes**, the existing fact table will be renamed to become the partition exchange table. Any current data will remain in this table.



© Copyright, WhereScape Software Ltd All rights reserved.

6. The parameters for the partitioned fact table are then prompted for and are described below:



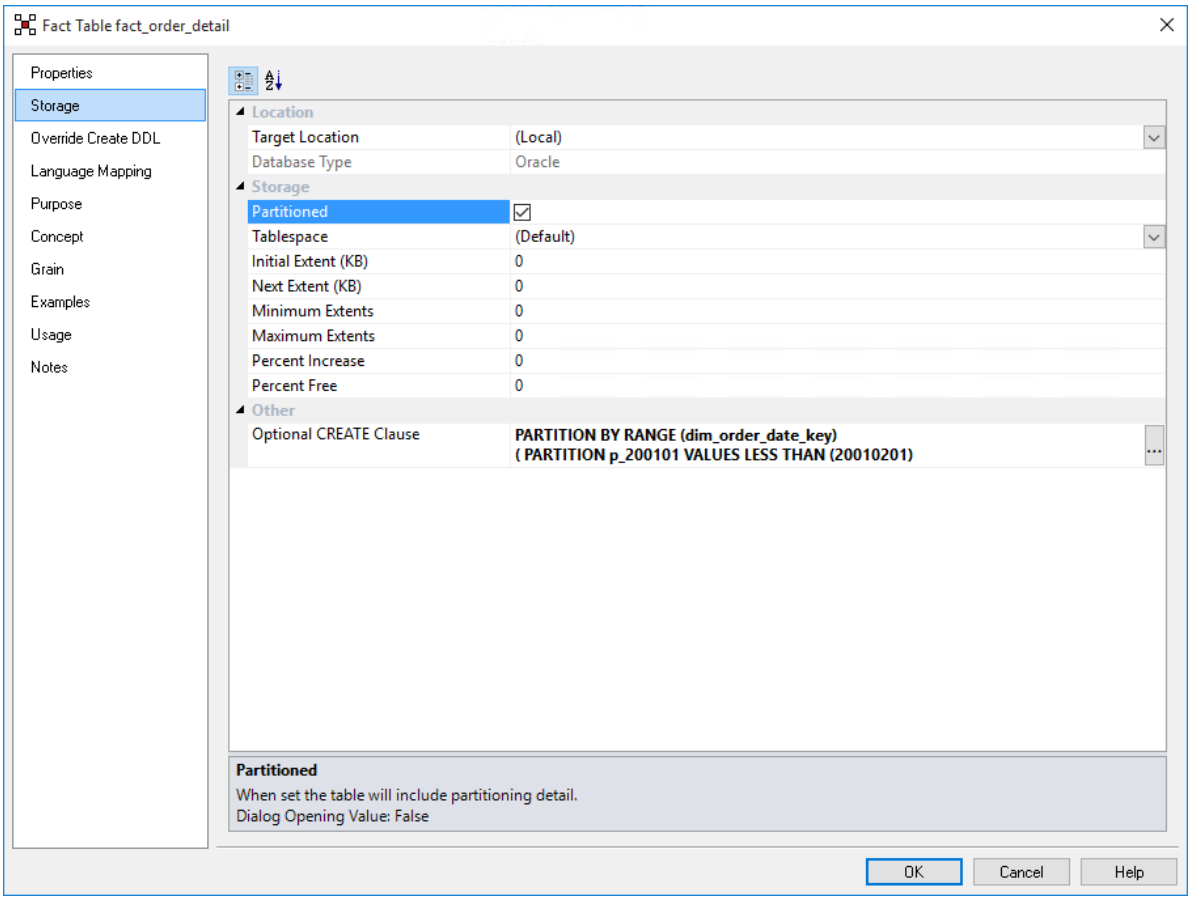
7. The partition exchange table name is prompted for. This table is identical to the fact table and is used to swap partitions to and from the fact table. Enter a **name** for the exchange partitioned table.
   - The fact table will be partitioned by the date dimension key. Select a **date dimension key** that will be used as the basis for the partitioning of the fact table. It is assumed that this key has the standard WhereScape format of YYYYMMDD.
   - Select a partition granularity of day, month or year along with the corresponding column in the date dimension.
     - For a **daily partition** a column with a date data type must be chosen. The date dimension key will actually be used for the partitioning but the date column is required.
     - For a **monthly partition** a column with the format YYYYMM must be chosen. In the standard WhereScape date dimension this is the column cal_month.
     - For a **yearly partition** a column with the format YYYY must be chosen. In the standard WhereScape date dimension this is the column cal_year.
   - The first and last partition should then be selected. The last partition is not that important as additional partitions will be created as required. Normally just select a partition a few on from the first. The first partition is however important as any data that is dated prior to the first partition will be loaded into this first partition. The partition must be entered in the format shown. For example monthly partitions require the format YYYYMM.
   - After all the relevant fields have been completed, click **OK** and the conversion process will populate the **Partition Function, Partition Scheme** and **Table Partition Scheme** fields of the table's storage properties.

Note

This field is limited in size, so if too many partitions are chosen only the first few will be added to this field. The other partitions will be created dynamically by the update procedure.

© Copyright, WhereScape Software Ltd All rights reserved.

8. Click the **OK** button on the table's **Storage** screen to begin the process of converting the fact table, creating the new fact table and building the new update procedure.
Examine the results screen to see what has been done.



The exchange partition table will now have any current data, to populate the fact table, copy the data from partition table to the fact table, drop and recreate fact index.
The date dimension key used to base the partitioning on is not allowed to have nulls. WhereScape RED has modified the table property, however the partition table needs to be validated and recreated to be valid.

Then the fact table can be processed.

© Copyright, WhereScape Software Ltd All rights reserved.

The partitioned fact table update procedure

The generated procedure will handle the creation of new partitions as required.

Any data for periods prior to the first partition is placed in the first partition. If data is received for a period after the last partition then new partitions are created.

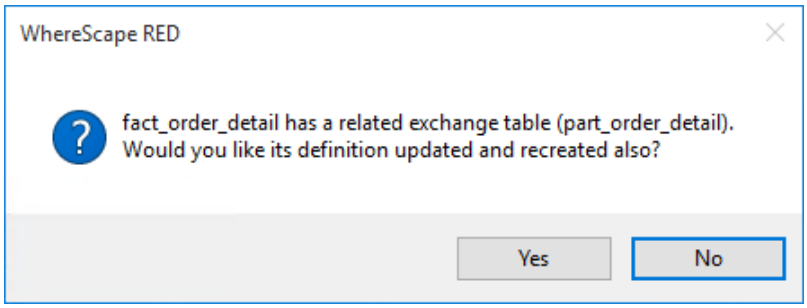
Empty partitions are created if necessary to skip periods where no data is present. For example, if we have a table partitioned by month and the latest partition is 200204 (April/2002). If data is then received for say September 2002 the update procedure will build partitions for May, June, July, August and September.

There is a variable at the start of the update procedure called v_max_skip_periods. This variable is by default set to 12. This defines the maximum number of continuous partitions we will leave empty.

From our previous example if our latest partition was April 2002 and we received data for July 2003 with no interim data then the update procedure will fail as it would have to skip 14 partitions. This check is present to prevent erroneous data from creating hundreds of partitions into the future.

Note

Recreating or Altering a partitioned fact table prompts with an offer to resync and recreate the related exchange table and indexes. This will resync the columns and indexes from the fact table to its exchange partition table so that partition swapping works correctly.



Generating the Partitioned Fact Update Procedure

Once a partitioned fact table has been defined in the metadata and created in the database, an update procedure can be generated to handle the update of the fact table.

Note

You can also generate an update procedure via a template, refer to Rebuilding Update Procedures for details.

Generating a Procedure

1. To generate a procedure, right-click on the fact table in the left pane and select **Properties**.
2. Click on the **Rebuild** button to start the process of generating the new procedure.
3. A series of prompts are displayed during the procedure generation based on the type of load information.

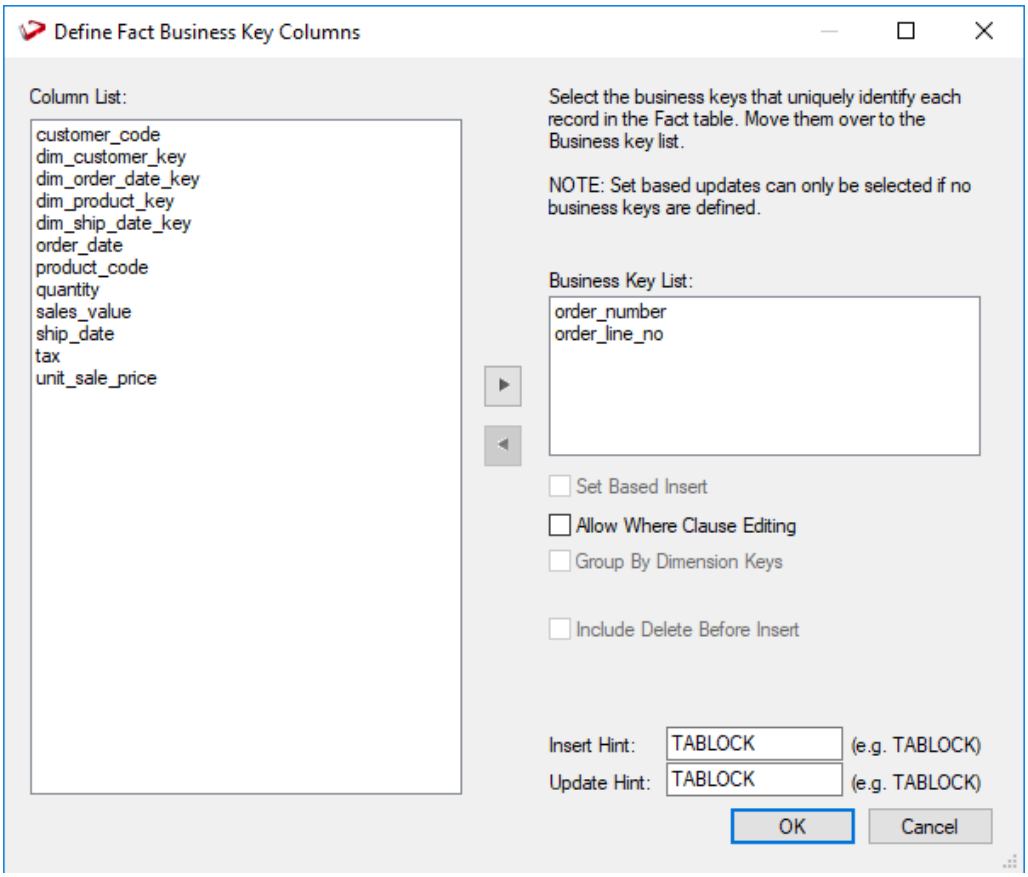
Define Fact Business Key Columns

The first dialog displayed when generating a partitioned fact table update procedure is the define fact business key columns dialog, asking for the business key that will uniquely identify each fact table record.

The source table from which the fact table is derived would normally have some form of unique constraint applied. In most cases this will be the business key.

In the example below the order_id and order_line_no are selected for the business key list.

© Copyright, WhereScape Software Ltd All rights reserved.



A business key can be made up of multiple columns, but it must provide a unique identifier. Where multiple columns separately uniquely identify rows in the fact table, choose one to act as the primary business key.

For example a source table may have a unique constraint on both a product code and a product description. Therefore the description as well as the code must be unique.

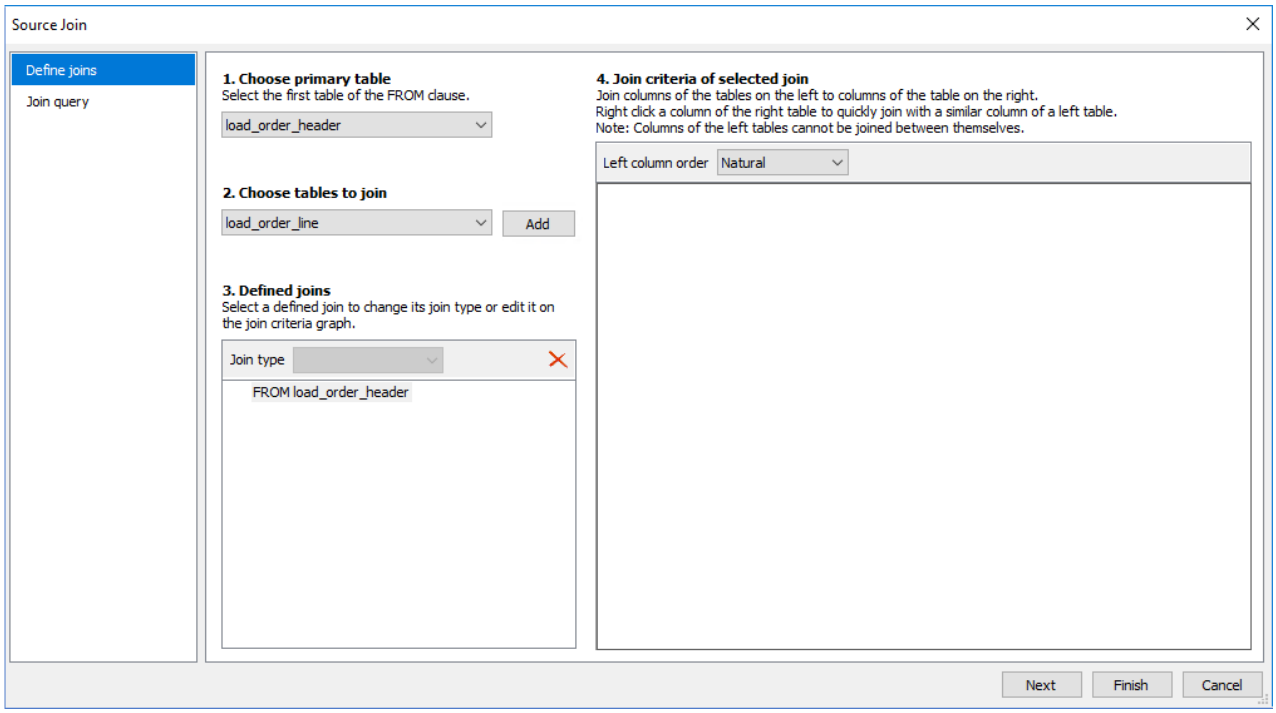
None of the columns chosen as the business key should ever contain a **NULL** value.

Fields	Description
Allow Where Clause Editing	If the **Allow Where Clause Editing** option is selected, then the next dialog displayed is the **Source Join** wizard, otherwise this is skipped.
Insert Hint	SQL Server only. Enter a database hint to be used in the INSERT statement.
Update Hint	SQL Server only. Enter a database hint to be used in the UPDATE statement.

© Copyright, WhereScape Software Ltd All rights reserved.

Source Join Wizard

This wizard is used to join source tables and add 'Where' clauses.



Source table joins should have been performed in the stage table, refer to **Generating the Staging Update Procedure** for details.

Since most of the real work is done in the stage table, this is all that is needed to build the Partitioned Fact Update Procedure.

Fact Table Column Properties

Each fact table column has a set of associated properties. The definition of each property is defined below:

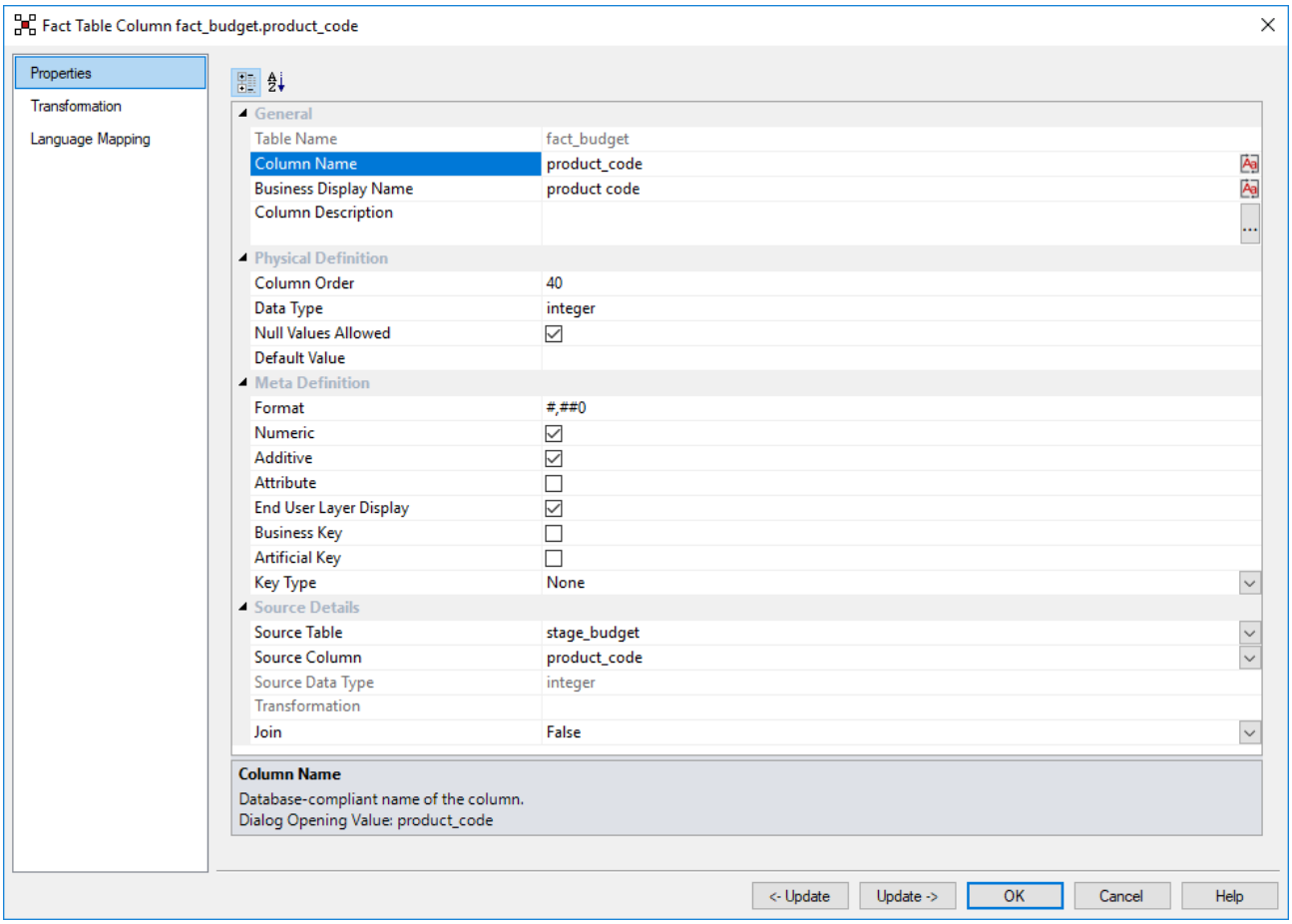
- If the **Column name** or **Data type** is changed for a column then the metadata will differ from the table as recorded in the database.
- Use the **Validate > Validate Table Create Status** menu option to compare the metadata to the table in the database.
  A context menu option **Alter Table** is available when your right click the table name, after the validate has completed. This option will alter the database table to match the metadata definition.

Tip

If a database table's definition is changed in the metadata, then the table needs to be altered in the database. Use the **Validate > Validate Table Create Status** to compare metadata definitions to physical database tables. The option also provides the ability to alter the database table, through a pop-up menu option from the validated table name.

© Copyright, WhereScape Software Ltd All rights reserved.

A sample **Properties** screen is as follows:



Tip

The two special update keys allow you to update the column and step either forward or backward to the next column's properties.
ALT-Left Arrow and **ALT-Right Arrow** can also be used instead of the two special update keys..

Fields	Description
Table Name	Database-compliant name of the table that contains the column. [Read-only].
Column Name	Database-compliant name of the column. Typically column-naming standards exclude the use of spaces etc. A good practice is to only use alphanumerics, and the underscore character. Changing this field alters the table's definition. **Note** A case conversion button on the right converts the text between different cases: UPPER CASE, Capitalized Case and lower case. The mode cycles to the next case in the sequence each conversion.
Business Display Name / Column Title	Name that the business uses to refer to the column, which is included in the RED-generated documentation and can be used in the end user layer of other tools. [Does NOT affect the physical database table]. As such it is a free form entry and any characters are valid. **Note** A case conversion button on the right converts the text between different cases: UPPER CASE, Capitalized Case and lower case. The mode cycles to the next case in the sequence each conversion.
Column Description	This field contains the description for the column. It may be a description from a business user's point of view. This field might additionally contain information on where and how the column was acquired. For example, if the column is sourced from

© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
	multiple tables or is a composite or derived column then this definition would normally describe the process used to populate the column. This field is used in the documentation and is available via the view **ws_admin_v_dim_col .** This field is also stored as a comment against the column in the database.
Column Order	Numeric value that controls the relative order of columns in the database create statement. The lowest numbered column will appear first in the table. Although this affects the physical table definition no action will be taken unless the table is re-created in the database. The columns can be re-numbered based on the existing order by choosing the **Respace Order Number** pop-up menu option when positioned over any column in the table. This action will number the columns in increments of 10 starting at 10. In addition to a simple change of the order field, the column order can be changed by first displaying the columns in the middle pane and then using drag and drop to move the columns around. This drag and drop process will automatically renumber the columns as required.
Data Type	Database-compliant data type that must be valid for the target database. Typical data types for Oracle are integer, number, char(), varchar2() and date. For SQL Server common types are integer, numeric, varchar() and datetime. See the database documentation for a description of the data types available. Changing this field alters the table's definition.
Null Values Allowed	Determines whether the table column can hold NULL values or whether a value is always mandatory.
Default Value	Initial value that is assigned to the column when a row is inserted into the table but no value is specified for the column.
Format	Optional format mask that can be used in end user tools. [Does NOT affect the physical database table]. As such it is a free form entry and any characters are valid. Typically format masks are only used on numeric fields. Example: #,###0.00. It is not worth the effort of populating this field unless it can be utilized by the end user tools in use.
Numeric	Indicates whether the table column holds values that are numeric. This is normally only relevant for fact tables. It does not affect the physical table definition, but rather provides input to the view ws_admin_v_dim_col which can be used to assist in the population of an end user tool's end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.
Additive	Indicates whether the table column holds values that are additive. This implies that the column can be summed when performing data grouping in a query. This is normally only relevant for fact tables. It does not affect the physical table definition, but rather provides input to the view ws_admin_v_dim_col which can be used to assist in the population of an end user tool's end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.
Attribute	Indicates whether the table column holds values that are descriptive, and/or are used for grouping/summing. An attribute is defined as a column that is non factual. For example we may have an order number, or a invoice number stored in the fact table. Such columns are considered attributes, rather than facts. This checkbox is therefore normally only relevant for fact tables. This checkbox does not affect the physical table definition, but rather provides input to the view **ws_admin_v_dim_col** which can be used to assist in the population of an end user tools end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.
End User Layer Display	Indicates whether the table column is available/visible to end users. If set the documentation will include the column in the glossary and in the user documentation. It is also used to decide what columns appear in the view ws_admin_v_dim_col. Typically columns such as the artificial key would not be enabled for end user display.
Business Key	Indicates whether the column is part of the BUSINESS key, which is defined during the update procedure build. [Normally maintained automatically]. Multiple columns can form the primary business key.

© Copyright, WhereScape Software Ltd All rights reserved.

<table>
<tr><th>Fields</th><th>Description</th></tr>
<tr><td>Key Type</td><td>Key type that is assigned and used when generating the table's update procedure and indexes. [Normally maintained automatically]. It can be altered here, but this should only be done if the consequences are fully understood and tested. The supported values are:
<table>
<tr><th>Key type</th><th>Meaning</th></tr>
<tr><td>0</td><td>The artificial key. Set when the key is added during drag and drop table generation.</td></tr>
<tr><td>1</td><td>Component of all business keys. Indicates that this column is used as part of any business key.</td></tr>
<tr><td>A</td><td>Indicates that the column is part of the primary business key. Set whenever a business key is defined as part of an update procedure generation.</td></tr>
<tr><td>B-Z</td><td>Indicates that the column is part of a secondary business key. Only used during index generation and not normally set.</td></tr>
</table></td></tr>
</table>

KPI Column Type	Only used by **KPI** fact tables. This field defines the column type for the KPI Fact Table. Refer to the KPI table creation section for more details on this field.
Source Table	Identifies the source table where the column's data comes from. This source table is normally a stage table within the data warehouse. If the column was sourced from multiple tables, then the normal practice is to record one of the tables in this field and a comment listing all of the other tables in the Source strategy field. This field is used when generating a procedure to update the fact table. It is also used in the track back diagrams and in the documentation.
Source Column	Identifies the source column where the column's data comes from. Such a column is normally a stage table column, which in turn may have been a transformation or the combination of multiple columns. This may also be a dimensional key where a dimension is being joined.
Source Data Type	Identifies the source column's data type. [Read-only].
Transformation	Refer to **Fact Table Column Transformations**. [Read-only].
Join	Indicates whether the table column is used in a table join. [Normally maintained automatically but can be optionally changed to override the default join logic used in the generated update procedure]. The **Source table** and **Source column** fields will provide the dimension table's side of the join. The options for this field are: False, True, Manual and Pre Join. • Setting this field to Manual changes the way the dimension table is looked up during the update procedure build. It allows you to join the dimension manually in the Cursor mapping dialog (where the 'Where' clause is built). • Setting this field to Pre Join activates the **Pre Join Source Table** field and allows you to select a table from the drop-down list.
Pre Join Source Table	Indicates the table from which the pre joined column was sourced. When the Join option is set to False, this field becomes inactive. When the Join option is set to True or Manual, this field is set to the current table name. When the Join option is set to Pre Join, then you can select the required table from the drop-down list.

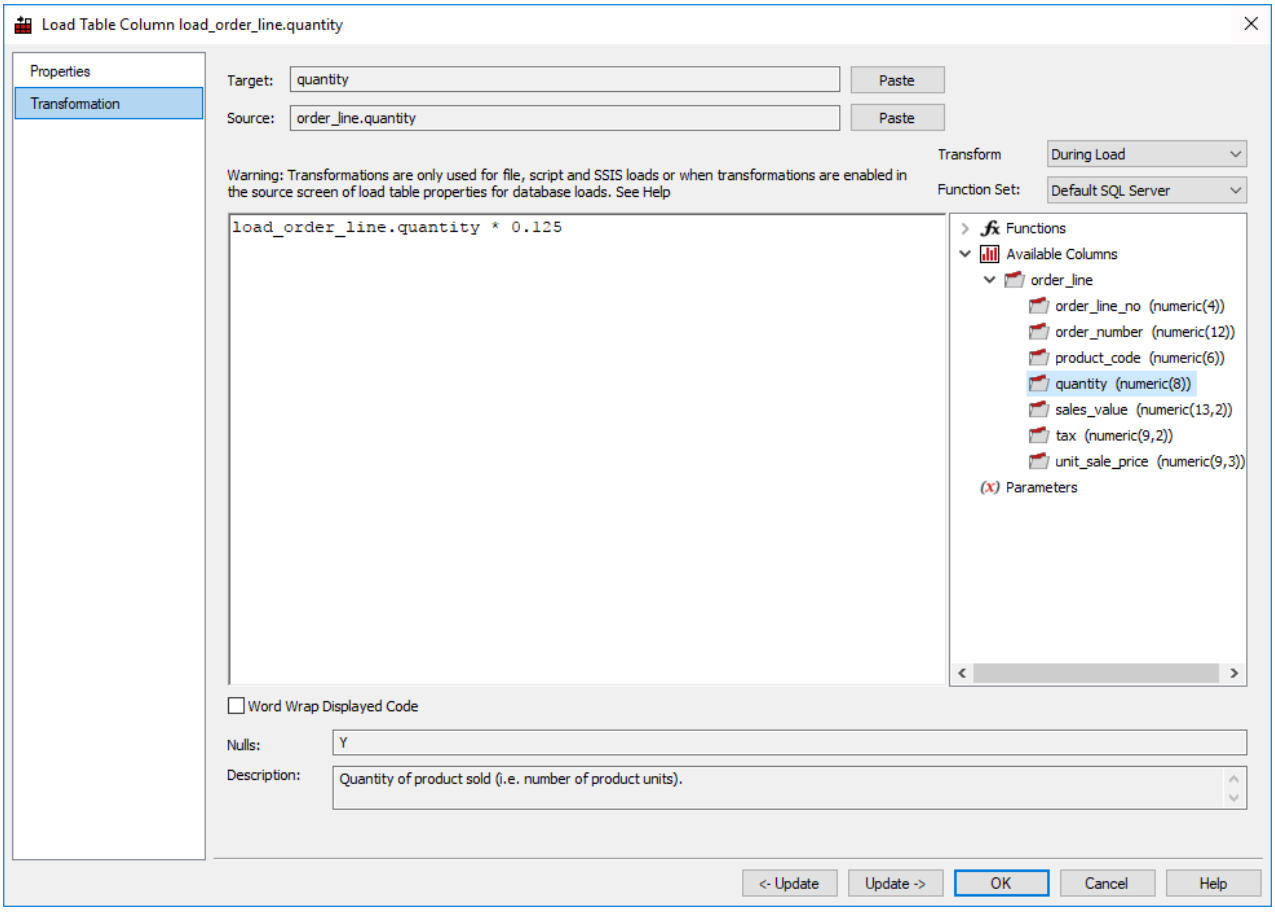
© Copyright, WhereScape Software Ltd All rights reserved.

Fact Table Column Transformations

Each fact table column can have a transformation associated with it. The transformation is included in the generated procedure and executed as part of the procedure update.
The transformation must therefore be a valid SQL construct that can be included in a **SELECT** statement.

For example, we could have a transformation of 'load_order_line.quantity * 0.125' to calculate a tax column of 12.5%.

Click the **Transformation** tab to enter a transformation.



Notes

- Transformations are only put into effect when the procedure is re-generated.
- **Microsoft Analysis Services 2005+ Tabular Mode Tables:** For Tabular Mode table column transformations, **Default DAX** is the only applicable Function Set for **after load** transformations.

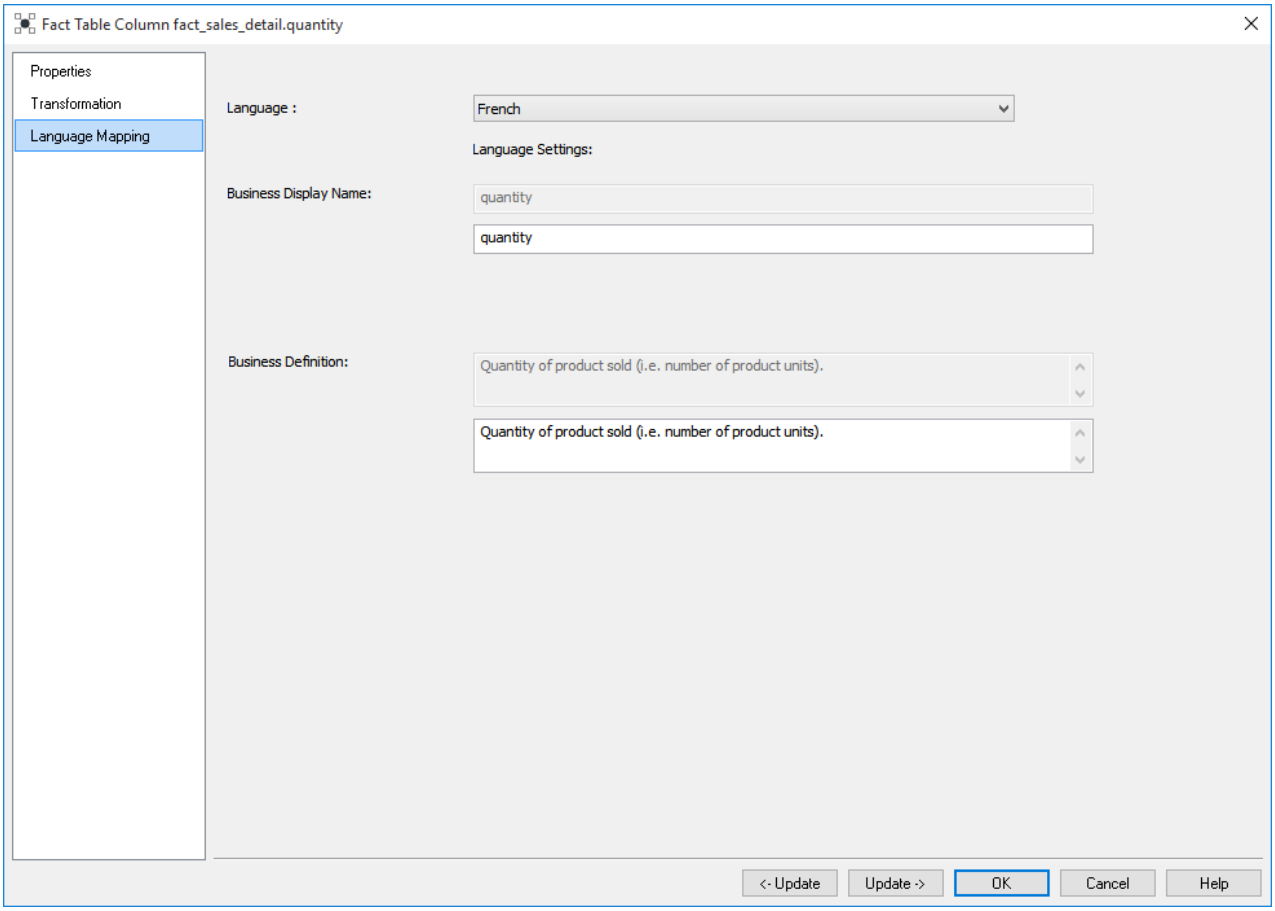
Refer to **Transformations** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

Fact Table Language Mapping

The Fact table Properties screen has a tab called **Language Mapping**.

Select the language from the drop-down list and then enter the translations for the **Business Display Name** and the **Description** in the chosen language.
The translations for these fields can then be pushed through into OLAP cubes.



© Copyright, WhereScape Software Ltd All rights reserved.

Aggregation

Two types of aggregate tables are discussed.

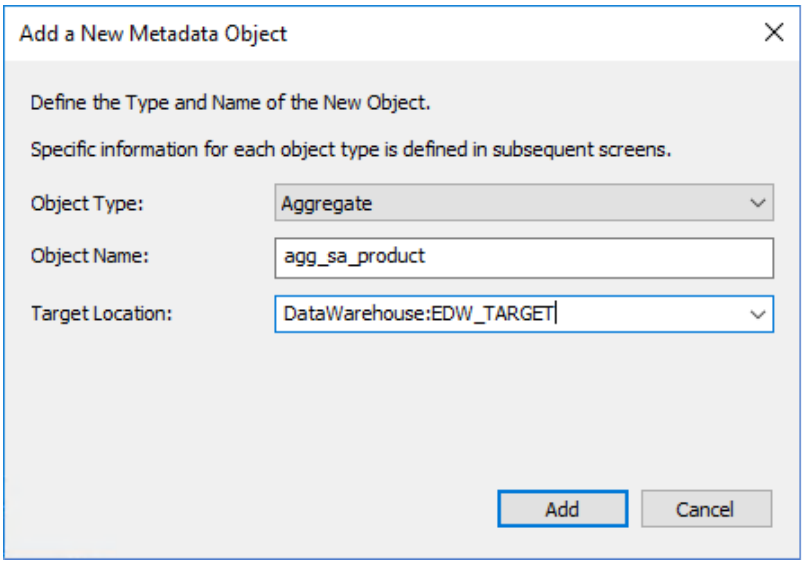
The first is where all non-additive facts and one or more dimensions are removed from a fact table. Typically this results in a smaller table that can answer a subset of the queries that could be posed against the fact table. This aggregate table still maintains full integrity to the remaining dimensions, and consequently reflects all changes to those dimensions.

The second type, we will call an aggregate summary, or summary table. This table includes additive measures and in some cases hierarchical elements of one or more of the dimensions providing a rolled-up summary of the fact table data. For example we may choose to deal at product group level rather than product SKU which is the granularity of the dimension.

When using the Oracle database the aggregate tables are seen as pre-built materialized views with reduced precision and query rewrite enabled.

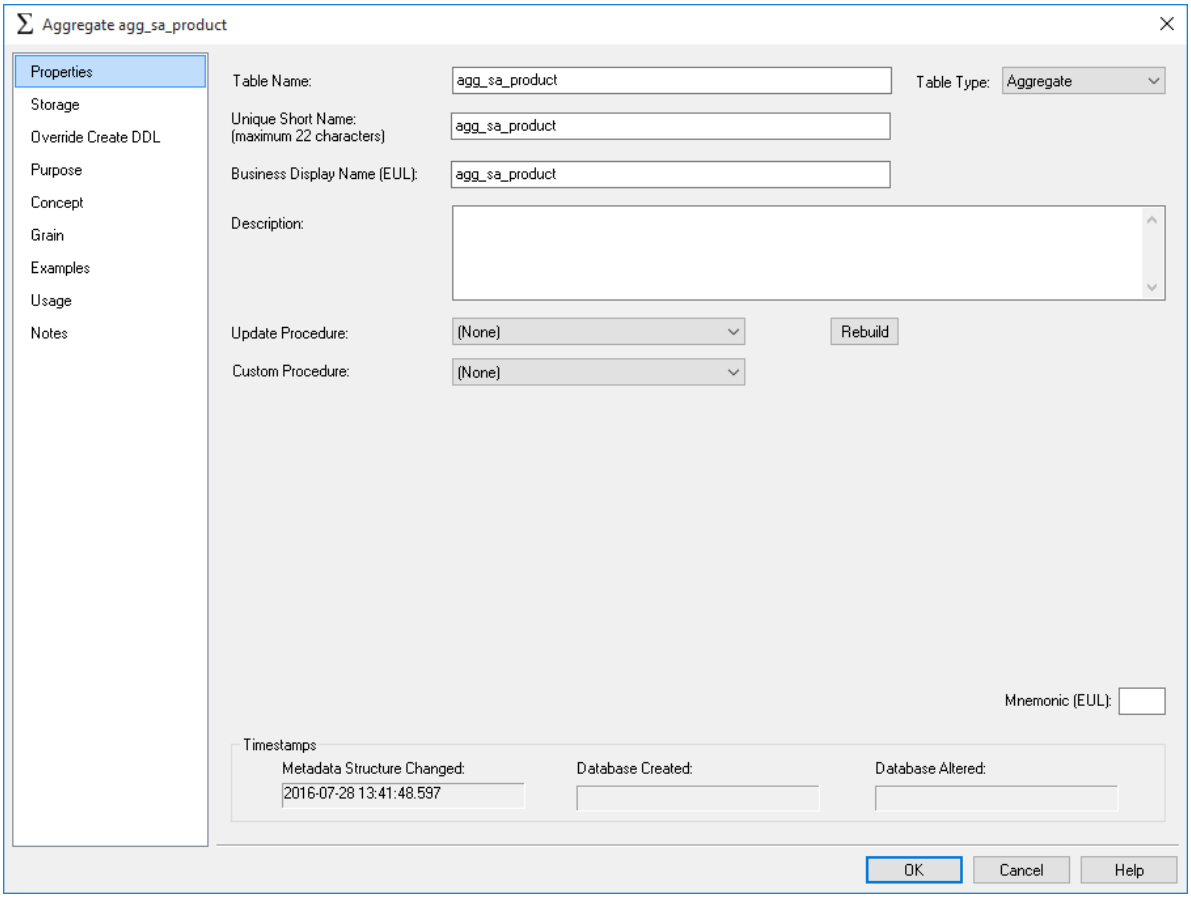
Creating an Aggregate Table

1. In the left pane double-click on the **aggregate** group to list the aggregates in the middle pane and set aggregates as the drop target.
2. From the Data Warehouse browse (right) pane drag a fact table into the middle pane and enter the aggregate name. Click **ADD**.

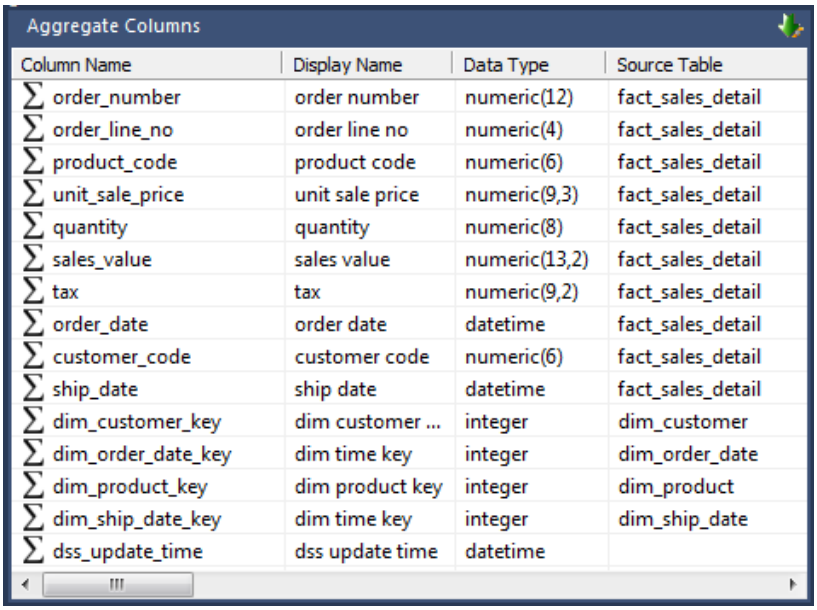


© Copyright, WhereScape Software Ltd All rights reserved.

3. The aggregate properties are displayed. Click **OK**.



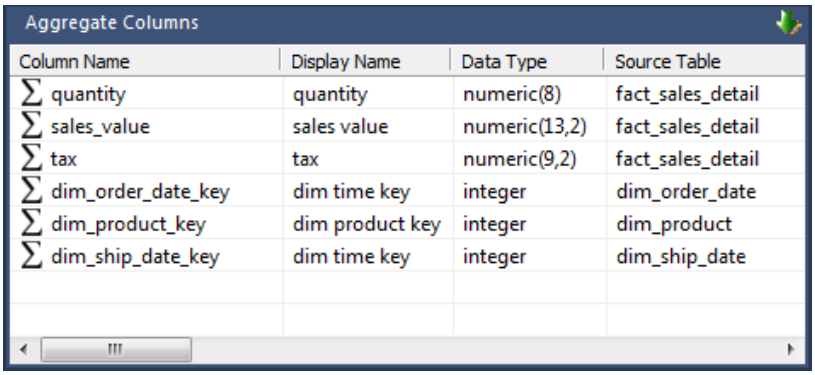
4. The list of columns in the Aggregate is displayed in the middle pane:



Column Name	Display Name	Data Type	Source Table
order_number	order number	numeric(12)	fact_sales_detail
order_line_no	order line no	numeric(4)	fact_sales_detail
product_code	product code	numeric(6)	fact_sales_detail
unit_sale_price	unit sale price	numeric(9,3)	fact_sales_detail
quantity	quantity	numeric(8)	fact_sales_detail
sales_value	sales value	numeric(13,2)	fact_sales_detail
tax	tax	numeric(9,2)	fact_sales_detail
order_date	order date	datetime	fact_sales_detail
customer_code	customer code	numeric(6)	fact_sales_detail
ship_date	ship date	datetime	fact_sales_detail
dim_customer_key	dim customer ...	integer	dim_customer
dim_order_date_key	dim time key	integer	dim_order_date
dim_product_key	dim product key	integer	dim_product
dim_ship_date_key	dim time key	integer	dim_ship_date
dss_update_time	dss update time	datetime	

5. Remove any columns that will not make sense at an aggregated level. For example, dss_fact_table_key, any business keys, any non-additive facts, any measures that relate to detail (e.g. unit price) and any dimension keys not required:

© Copyright, WhereScape Software Ltd All rights reserved.



Note

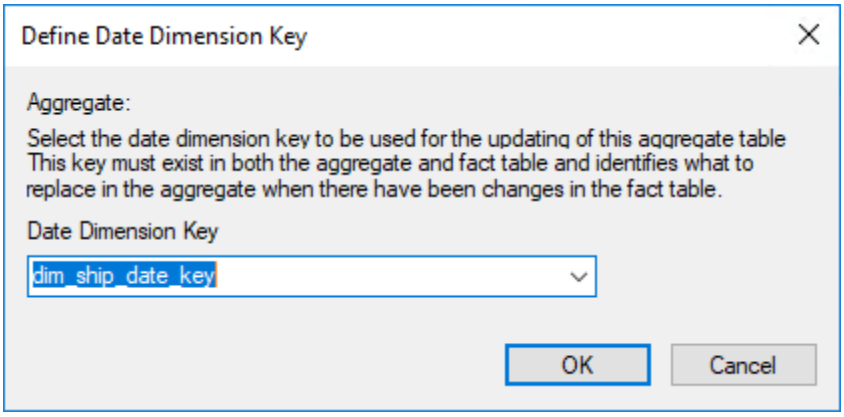
All aggregate table columns (other than dss columns) should always have a source table specified, even if a column transformation is in use.

6. Create the aggregate table in the database by right-clicking on the aggregate and selecting **Create(ReCreate)**.
7. Create a procedure to update the aggregate by right-clicking on the aggregate, selecting **Properties** and selecting **(Build Procedure...)** in the Update Procedure field.

Note

WhereScape RED generated update procedures for aggregates are incremental. Incremental updates are based on a date dimension key and number of look back days. The aggregate update process looks at any records that have been updated in the fact table in the last 7 days (by default)..

To support this, a date dimension key must be selected. The columns chosen must be in both the source fact table and the aggregate. Select this column and click **OK**.

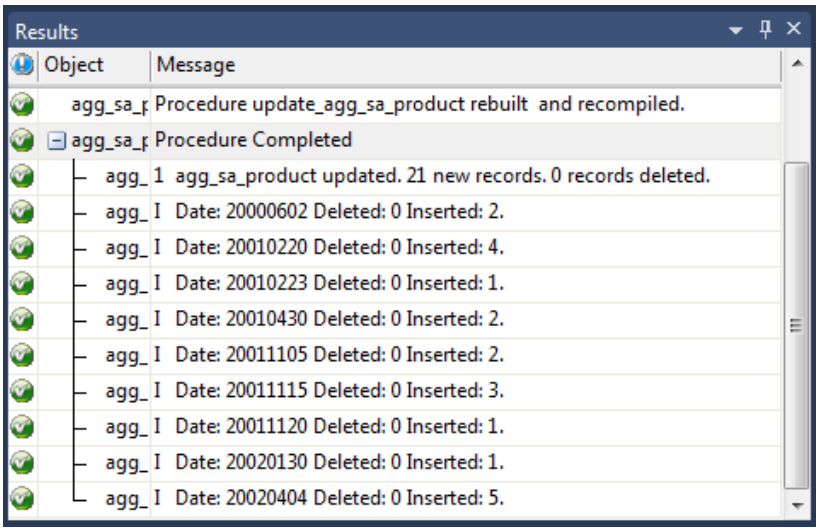


8. Update the table by right-clicking and choosing **Execute Update Procedure.**

Note

Any column transformations added to a measure column of an aggregate table must always include an aggregate function, usually **SUM**. For example, an ISNULL added to forecast_quantity should be entered as: SUM(ISNULL(forecast_quantity,0)).

© Copyright, WhereScape Software Ltd All rights reserved.



- **Oracle**: When the procedure runs it first removes the materialized view, then updates any changes from the fact table into the aggregate. It then re-establishes the materialized view, enabling query rewrite.
- **DB2:** When the procedure runs it first removes the materialized query table, then updates any changes from the fact table into the aggregate. It then re-establishes the materialized query table, enabling query rewrite.

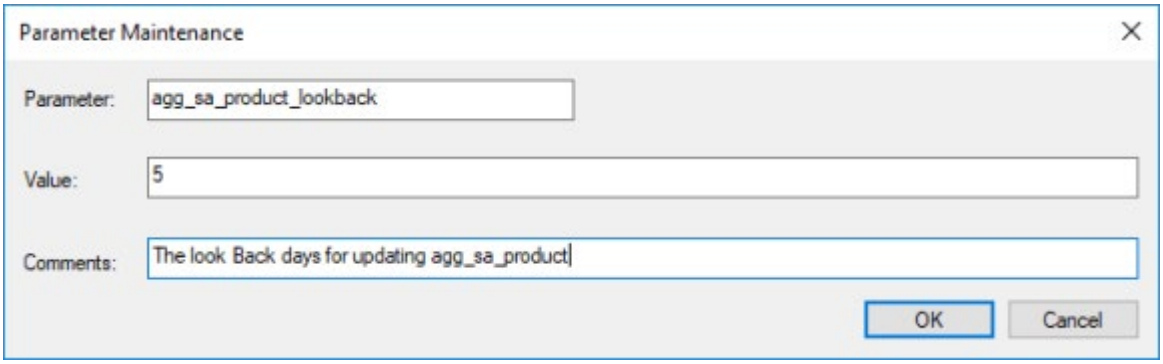
Change an Aggregate's Default Look Back Days

WhereScape RED generate update procedures for aggregates that are incremental. Incremental updates are based on a date dimension key and number of look back days.

The aggregate update process looks at any records that have been updated in the fact table in the last number of look back days.

The default number of look back days is 7. The default is set in each update procedure. To change the number of look back days for an aggregate table, create a WhereScape RED parameter called AggregateName_lookback and enter the required number of look back days as the parameter value.

For example, add the following parameter to change the look back days to 5 for agg_sa_product:



Refer to **Parameters** for more information on WhereScape RED parameters.

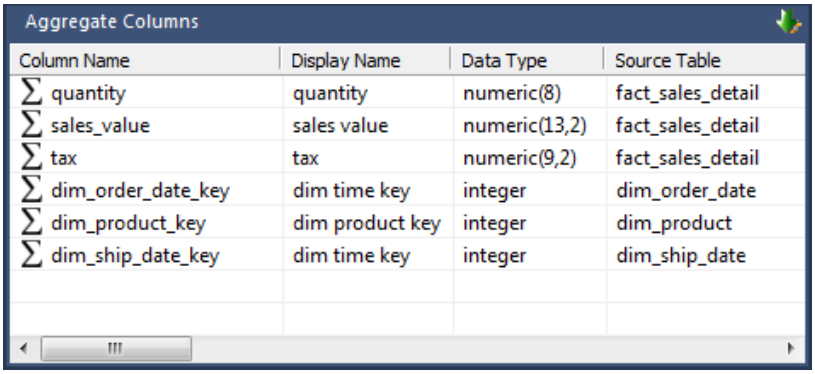
Creating an Aggregate Summary Table

The creation of a summary table proceeds initially in the same way as an aggregate table.

1. In the left pane double-click the **Aggregate** group to list the aggregates in the middle pane and set aggregates as the drop target.
2. From the Data Warehouse browse (right) pane drag a fact table into the middle pane. Remove any columns that will not make sense at an aggregated level. For example, dss_fact_table_key, any business keys, any non-additive facts, any measures that relate to detail (e.g. unit price).

© Copyright, WhereScape Software Ltd All rights reserved.

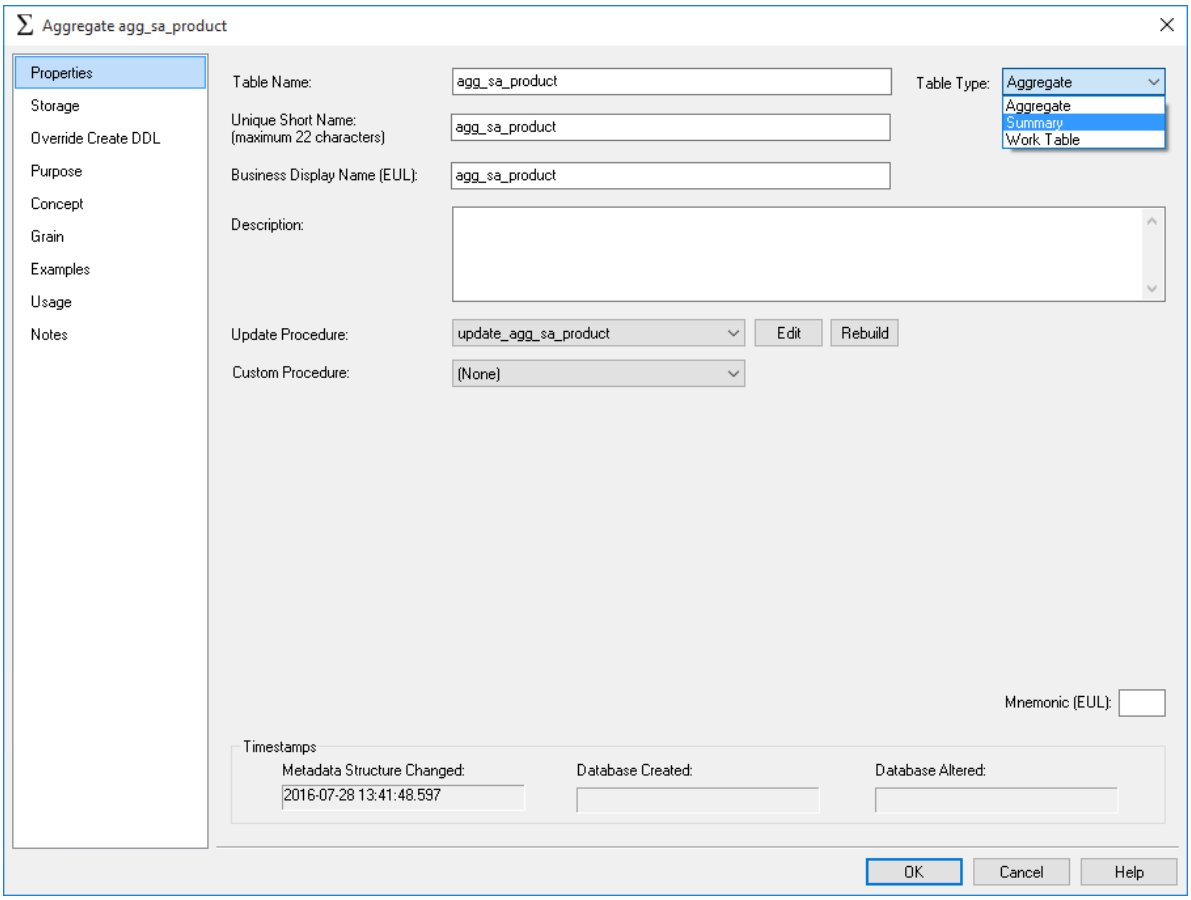
3. Drag over columns from dimensions that are linked to the fact table. Delete the dimension keys to allow a rollup to the level of the dimension elements.



Note

All aggregate summary table columns (other than dss columns) should always have a source table specified, even if a column transformation is in use.

4. In the **Properties** of the aggregate table change the **Table Type** to **Summary**:

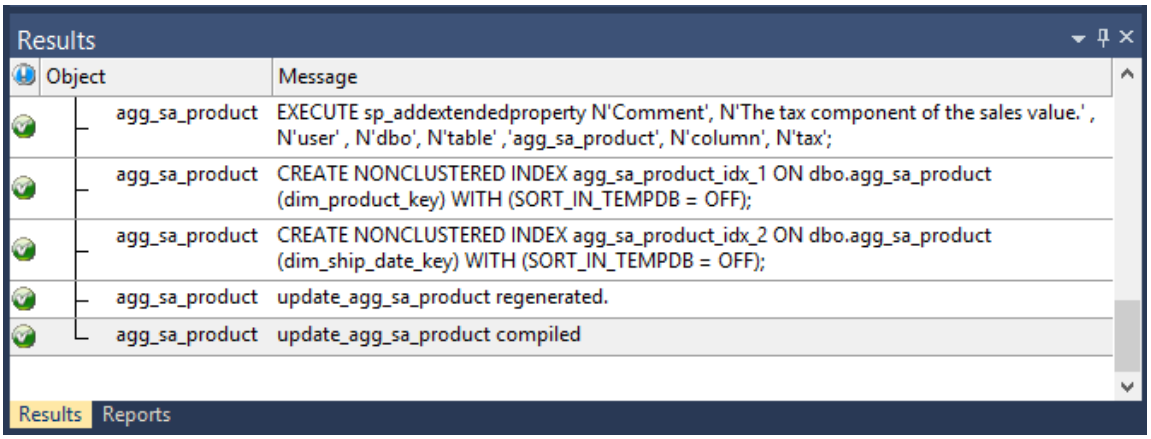


5. Create the aggregate summary table in the database by right-clicking on the aggregate and selecting **Create(ReCreate)**.
6. Create a procedure to update the aggregate summary by right-clicking the aggregate, selecting **Properties** and selecting **(Build Procedure...)** in the **Update Procedure** field. The aggregate summary table is totally rebuilt each time the procedure is executed.

Note

Any column transformations added to a measure column of an aggregate summary table must always include an aggregate function, usually **SUM**. For example, an ISNULL added to forecast_quantity should be entered as: SUM(ISNULL(forecast_quantity,0)).

© Copyright, WhereScape Software Ltd All rights reserved.



© Copyright, WhereScape Software Ltd All rights reserved.

Aggregate Table Column Properties

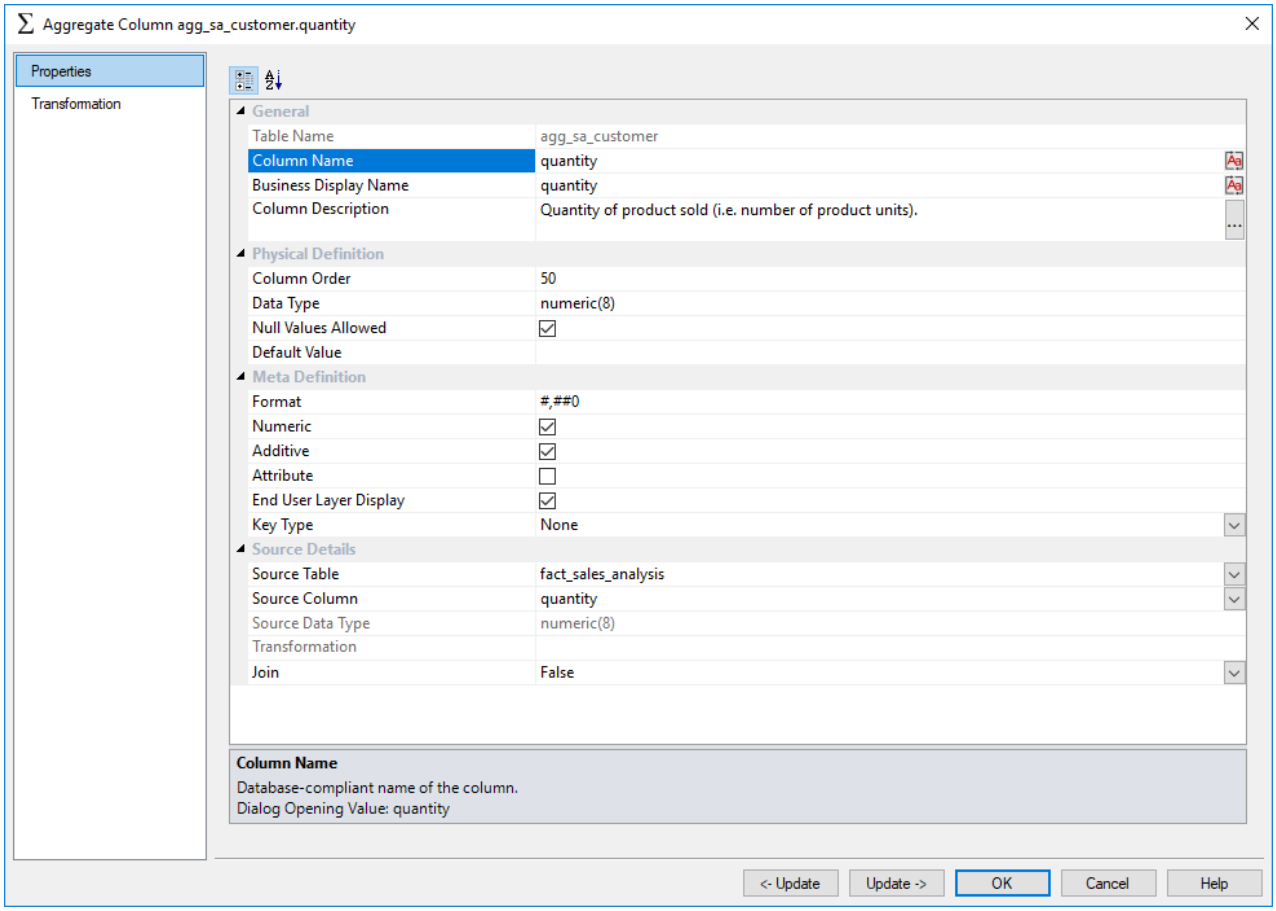
Each aggregate table column has a set of associated properties. The definition of each property is defined below:

If the **Column name** or **Data type** is changed for a column, then the metadata will differ from the table as recorded in the database. Use the **Validate > Validate Table Create Status** menu option to compare the metadata to the table in the database. A right click menu option of **Alter Table** is available when positioned on the table name after the validate has completed. This option will alter the database table to match the metadata definition.

Tip

If a database table's definition is changed in the metadata then the table will need to be altered in the database. Use the **Validate > Validate Table Create Status** to compare metadata definitions to physical database tables. The option also provides the ability to alter the database table, through a pop-up menu option from the validated table name.

A sample **Properties** screen:



The two special update keys allow you to update the column and step either forward or backward to the next column's properties. **ALT-Left Arrow** and **ALT-Right Arrow** can also be used instead of the two special update keys.

General

Fields	Description
Table Name	Database-compliant name of the table that contains the column. [Read-only].
Column Name	Database-compliant name of the column. Typically column-naming standards exclude the use of spaces etc. A good practice is to only use alphanumerics, and the underscore character. Changing this field alters the table's definition.

© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
	Note A case conversion button on the right converts the text between different cases: UPPER CASE, Capitalized Case and lower case. The mode cycles to the next case in the sequence each conversion.
Business Display Name	Name that the business uses to refer to the column, which is included in the RED-generated documentation and can be used in the end user layer of other tools. [Does NOT affect the physical database table]. As such it is a free form entry and any characters are valid. **Note** A case conversion button on the right converts the text between different cases: UPPER CASE, Capitalized Case and lower case. The mode cycles to the next case in the sequence each conversion.
Column Description	This field contains the description for the column. It may be a description from a business user's point of view. This field might additionally contain information on where and how the column was acquired. For example if the column is sourced from multiple tables or is a composite or derived column then this definition would normally describe the process used to populate the column. This field is used in the documentation and is available via the view **ws_admin_v_dim_col .** This field is also stored as a comment against the column in the database.

Physical Definition

Fields	Description
Column Order	Numeric value that controls the relative order of columns in the database create statement. The lowest numbered column will appear first in the table. Although this affects the physical table definition no action will be taken unless the table is re-created in the database. The columns can be re-numbered based on the existing order by choosing the **Respace Order Number** pop-up menu option when positioned over any column in the table. This action will number the columns in increments of 10 starting at 10. In addition to a simple change of the order field, the column order can be changed by first displaying the columns in the middle pane and then using drag and drop to move the columns around. This drag and drop process will automatically renumber the columns as required.
Data Type	Database-compliant data type that must be valid for the target database. Typical data types for Oracle are integer, number, char(), varchar2() and date. For SQL Server common types are integer, numeric, varchar() and datetime. See the database documentation for a description of the data types available. Changing this field alters the table's definition.
Null Values Allowed	Determines whether the table column can hold NULL values or whether a value is always mandatory.
Default Value	Initial value that is assigned to the column when a row is inserted into the table but no value is specified for the column.

Meta Definition

Fields	Description
Format	Optional format mask that can be used in end user tools. [Does NOT affect the physical database table]. As such it is a free form entry and any characters are valid. Typically format masks are only used on numeric fields. Example: #,###0.00. It is not worth the effort of populating this field unless it can be utilized by the end user tools in use.

© Copyright, WhereScape Software Ltd All rights reserved.

<table>
<tr><th>Fields</th><th>Description</th></tr>
<tr><td>Numeric</td><td>Indicates whether the table column holds values that are numeric. This is normally only relevant for fact tables. It does not affect the physical table definition, but rather provides input to the view ws_admin_v_dim_col which can be used to assist in the population of an end user tool's end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.</td></tr>
<tr><td>Additive</td><td>Indicates whether the table column holds values that are additive. This implies that the column can be summed when performing data grouping in a query. This is normally only relevant for fact tables. It does not affect the physical table definition, but rather provides input to the view ws_admin_v_dim_col which can be used to assist in the population of an end user tool's end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.</td></tr>
<tr><td>Attribute</td><td>Indicates whether the table column holds values that are descriptive, and/or are used for grouping/summing. An attribute is defined as a column that is non factual. For example we may have an order number, or a invoice number stored in the fact table. Such columns are considered attributes, rather than facts. This checkbox is therefore normally only relevant for fact tables. This checkbox does not affect the physical table definition, but rather provides input to the view ws_admin_v_dim_col which can be used to assist in the population of an end user tools end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.</td></tr>
<tr><td>End User Layer Display</td><td>Indicates whether the table column is available/visible to end users. If set the documentation will include the column in the glossary and in the user documentation. It is also used to decide what columns appear in the view ws_admin_v_dim_col. Typically columns such as the artificial key would not be enabled for end user display.</td></tr>
<tr><td>Key Type</td><td>Key type that is assigned and used when generating the table's update procedure and indexes. [Normally maintained automatically]. It can be altered here, but this should only be done if the consequences are fully understood and tested. The supported values are:
<table>
<tr><th>Key type</th><th>Meaning</th></tr>
<tr><td>0</td><td>The artificial key. Set when the key is added during drag and drop table generation.</td></tr>
<tr><td>1</td><td>Component of all business keys. Indicates that this column is used as part of any business key. For example: By default the dss_source_system_key is added to every dimension table. It is considered part of any lookup on that table and has the key type set to 1. Set when the column is added during drag and drop table generation.</td></tr>
<tr><td>2</td><td>Indicates that this column is a dimensional join. Used on fact tables to indicate the dimension keys. Results in bitmap indexes being built for the columns. Set during the update procedure generation for a fact table, based on information from the staging table.</td></tr>
<tr><td>3</td><td>Slowly changing column indicator. Used on dimension tables to indicate that the column is being managed as a slowly changing column within the context of a slowly changing dimension. Set when a column is identified during the dimension update procedure generation.</td></tr>
<tr><td>4</td><td>Previous value column indicator. Used on dimension tables to indicate that the column is being managed as a previous value column. The source column identifies the parent column. Set during the dimension creation.</td></tr>
<tr><td>A</td><td>Indicates that the column is part of the primary business key. Set whenever a business key is defined as part of an update procedure generation.</td></tr>
<tr><td>B-Z</td><td>Indicates that the column is part of a secondary business key.</td></tr>
</table>
</td></tr>
</table>

© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description	
		Only used during index generation and not normally set.

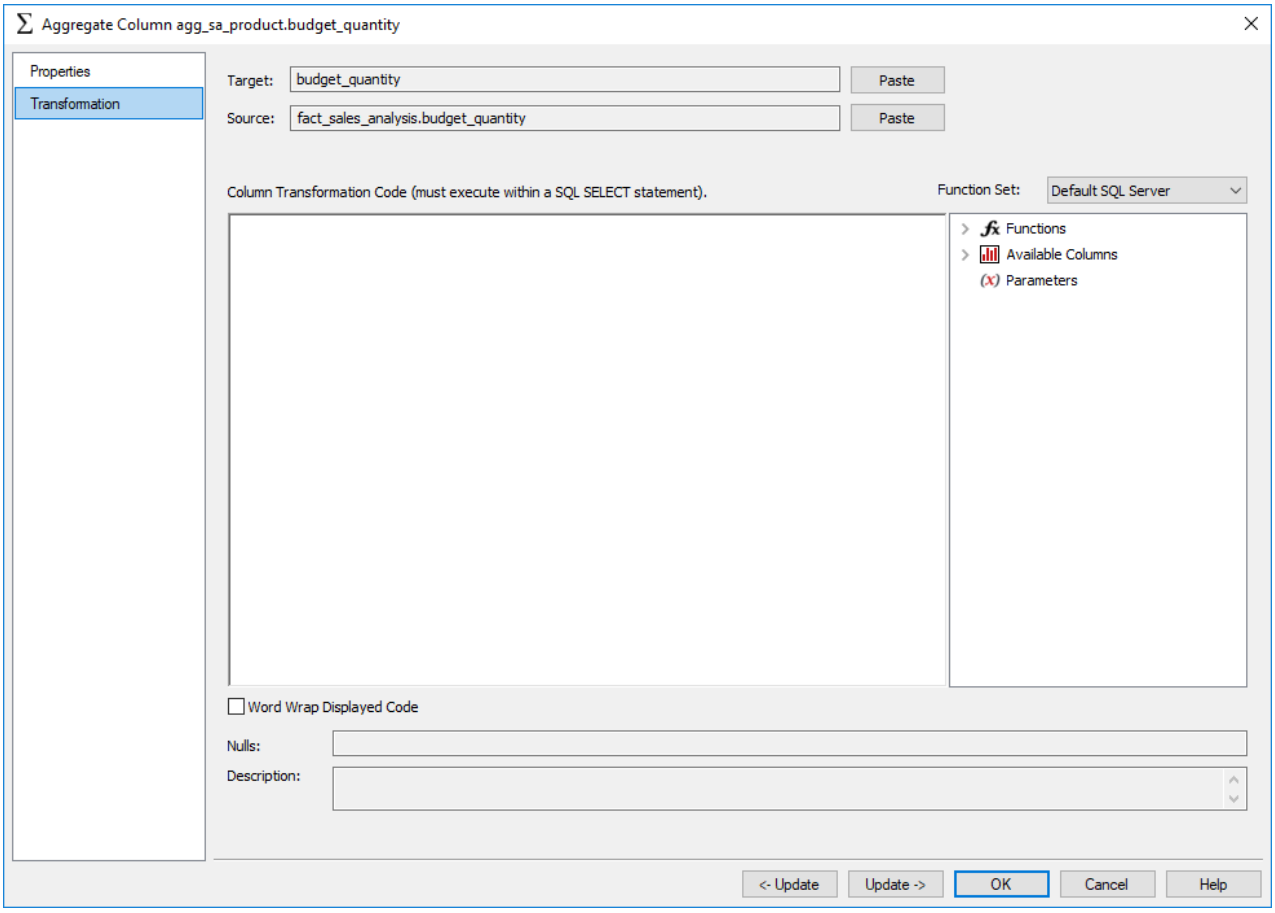
Source Details

Fields	Description
Source Table	Identifies the source table where the column's data comes from. This source table is normally a fact table or a dimension table within the data warehouse. If the column was sourced from multiple tables, then the normal practice is to record one of the tables in this field and a comment listing all of the other tables in the **Source strategy** field. This field is used when generating a procedure to update the aggregate table. It is also used in the track back diagrams and in the documentation.
Source Column	Identifies the source column where the column's data comes from. Such a column is normally a fact table column or a dimension table column, which in turn may have been a transformation or the combination of multiple columns.
Source Data Type	Identifies the source column's data type. [Read-only].
Transformation	Refer to **Aggregate Table Column Transformations**. [Read-only].
Join	Indicates whether the table column is used in a table join. [Normally maintained automatically but can be optionally changed to override the default join logic used in the generated update procedure]. The **Source Table** and **Source Column** fields will provide the dimension table's side of the join. The options for this field are: False, True, Manual and Pre Join. • Setting this field to Manual changes the way the dimension table is looked up during the update procedure build. It allows you to join the dimension manually in the **Source Join** wizard (used to build the 'Where' clause). • Setting this field to Pre Join activates the **Pre Join Source Table** field and allows you to select a table from the drop-down list.
Pre Join Source Table	Indicates the table from which the pre joined column was sourced. When the Join option is set to False, this field becomes inactive. When the Join option is set to True or Manual, this field is set to the current table name. When the Join option is set to Pre Join, then you can select the required table from the drop-down list.

© Copyright, WhereScape Software Ltd All rights reserved.

Aggregate Table Column Transformations

Each aggregate table column can have a transformation associated with it. The transformation will be included in the generated procedure and will be executed as part of the procedure update. The transformation must therefore be a valid SQL construct that can be included in a **Select** statement. For example we could have a transformation of 'load_order_line.quantity * 0.125' to calculate a tax column of 12.5%. Click the **Transformation** tab to enter a transformation.



Note

Transformations are only put into effect when the procedure is re-generated.

Refer to **Transformations** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

Views

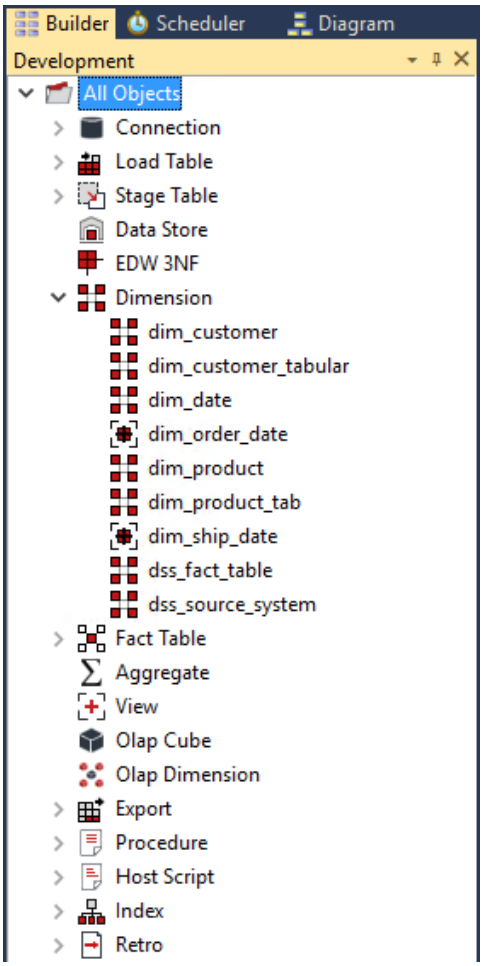
Views are normally created from Dimensions. These Views are then used to define Dimensions in a Fact table where the Dimension appears multiple times. A classic example is the date dimension. To facilitate the creation of the Fact table we create two dimension views to allow the date dimension to be joined to the Fact table twice.

Views can also be created from any other table type. By default, views are displayed as part of the Dimension object group. If Fact table views or other views are being used, then views can be displayed as a separate object type which can be configured from **Tools > Options > User Preferences > Look and Feel > General**.

Dimension Views

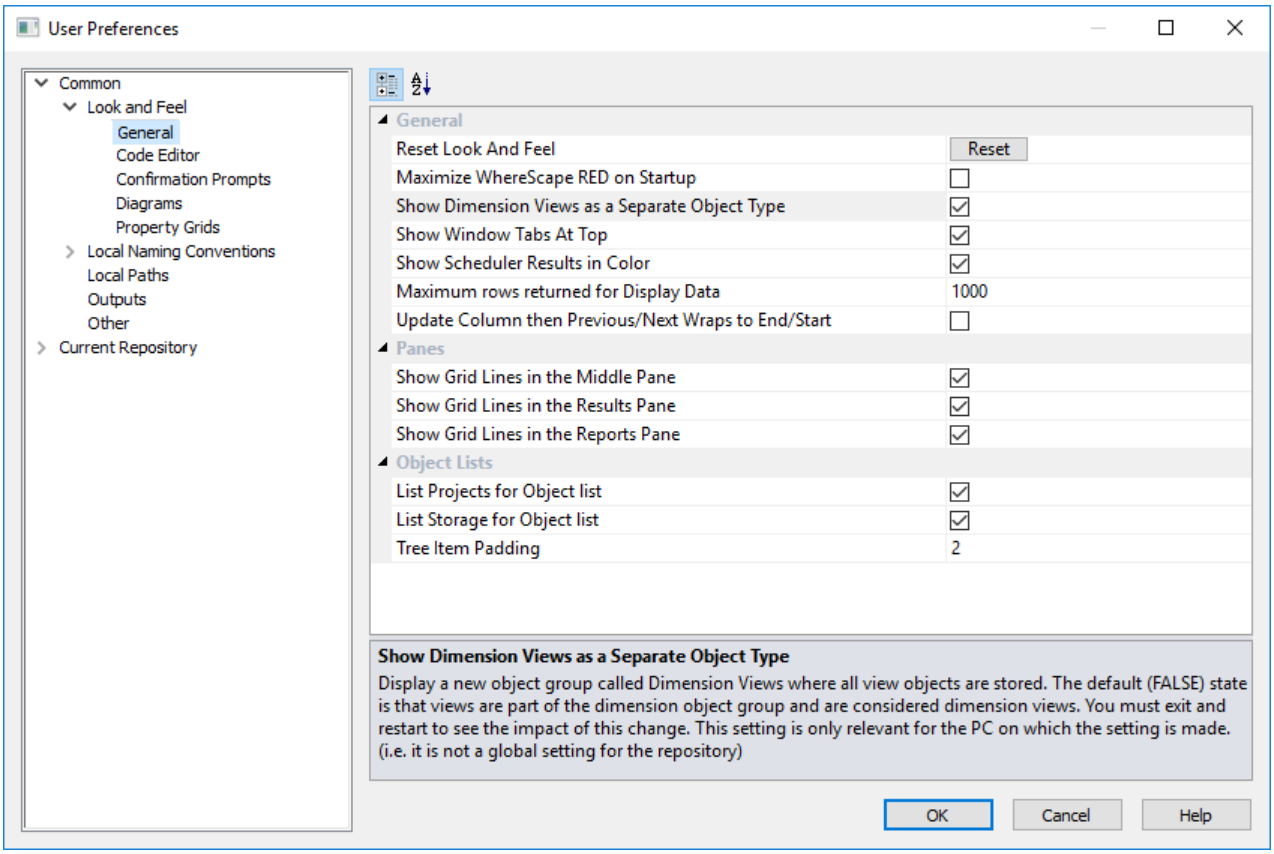
A Dimension view is a database view of a Dimension table. It may be a full or partial view. It is typically used in cases such as date dimensions, where multiple date dimensions exist for one Fact table.
In many data warehouses Dimension views are built as part of the end user layer but creating them in the data warehouse means they are available regardless of the end user tools used.

Views are displayed in WhereScape RED either as part of the **Dimension** object group or in a separate object group—**Dimension View**.
The default is to view Dimensions and Dimension Views together:



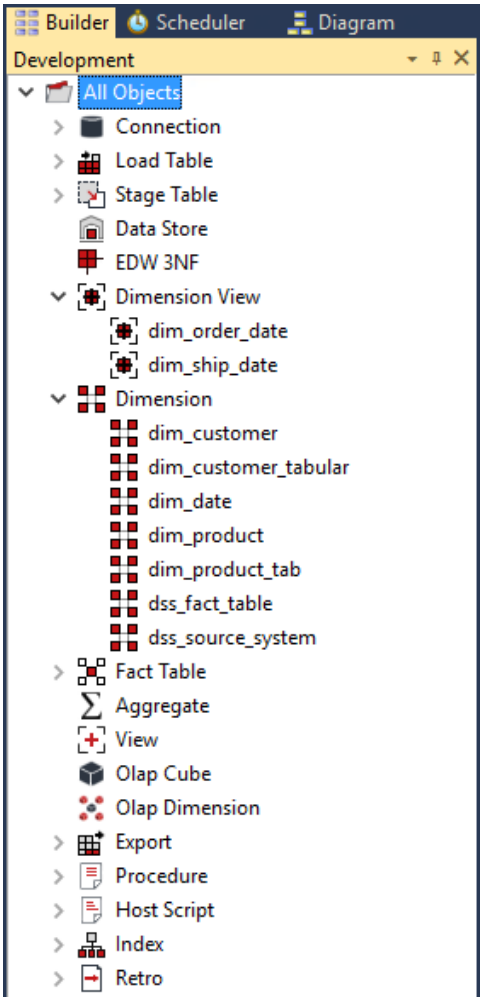
© Copyright, WhereScape Software Ltd All rights reserved.

This visualization option is set from the **Tools > User Preferences > Look and Feel** menu item:



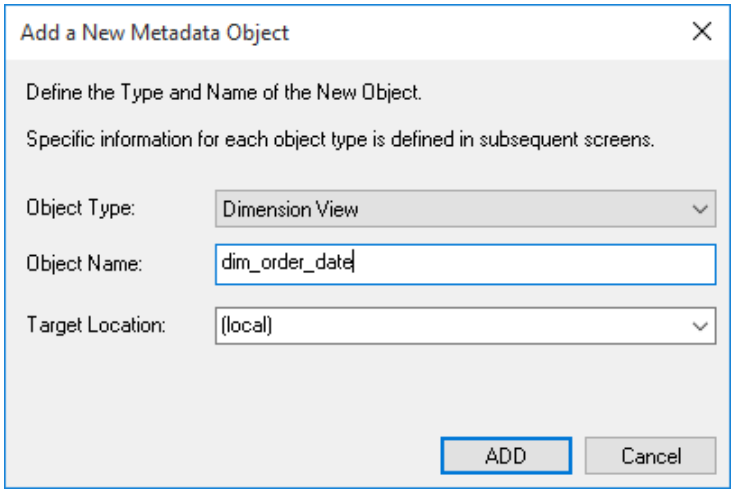
© Copyright, WhereScape Software Ltd All rights reserved.

This example shows **Dimensions** and **Dimension Views** as separate object types:



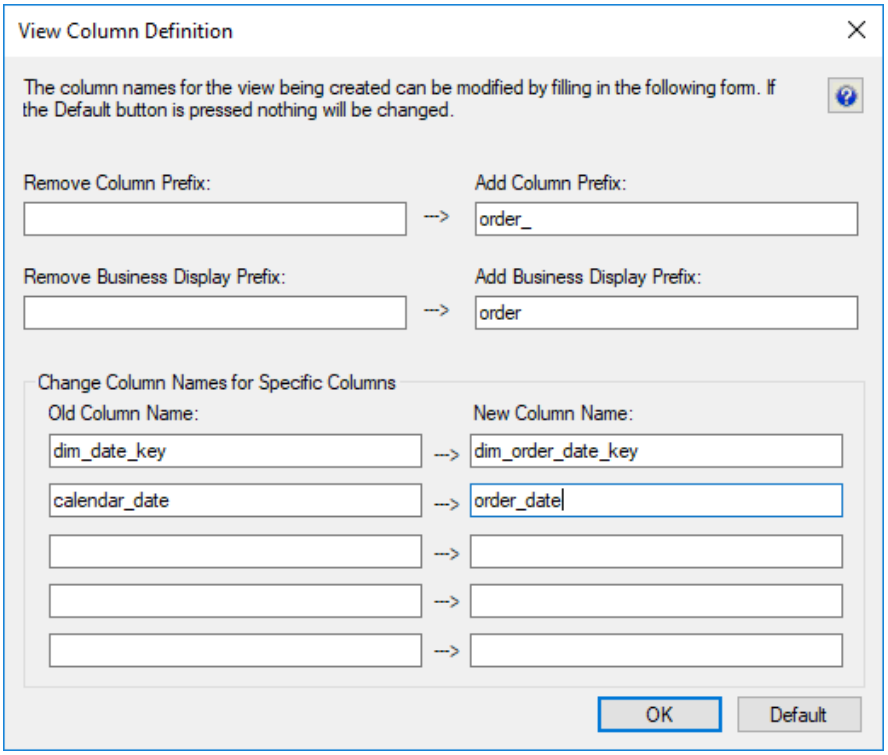
Creating a Dimension View:

1. Double-click Dimension (or Dimension View) in the left pane.
2. Browse the data warehouse in the right pane.
3. Drag a table from the right pane into the middle pane.
    - The **Add New Metadata Object** window defaults the object type to a Dimension view
    - Change the view name as required and then click **Add**.

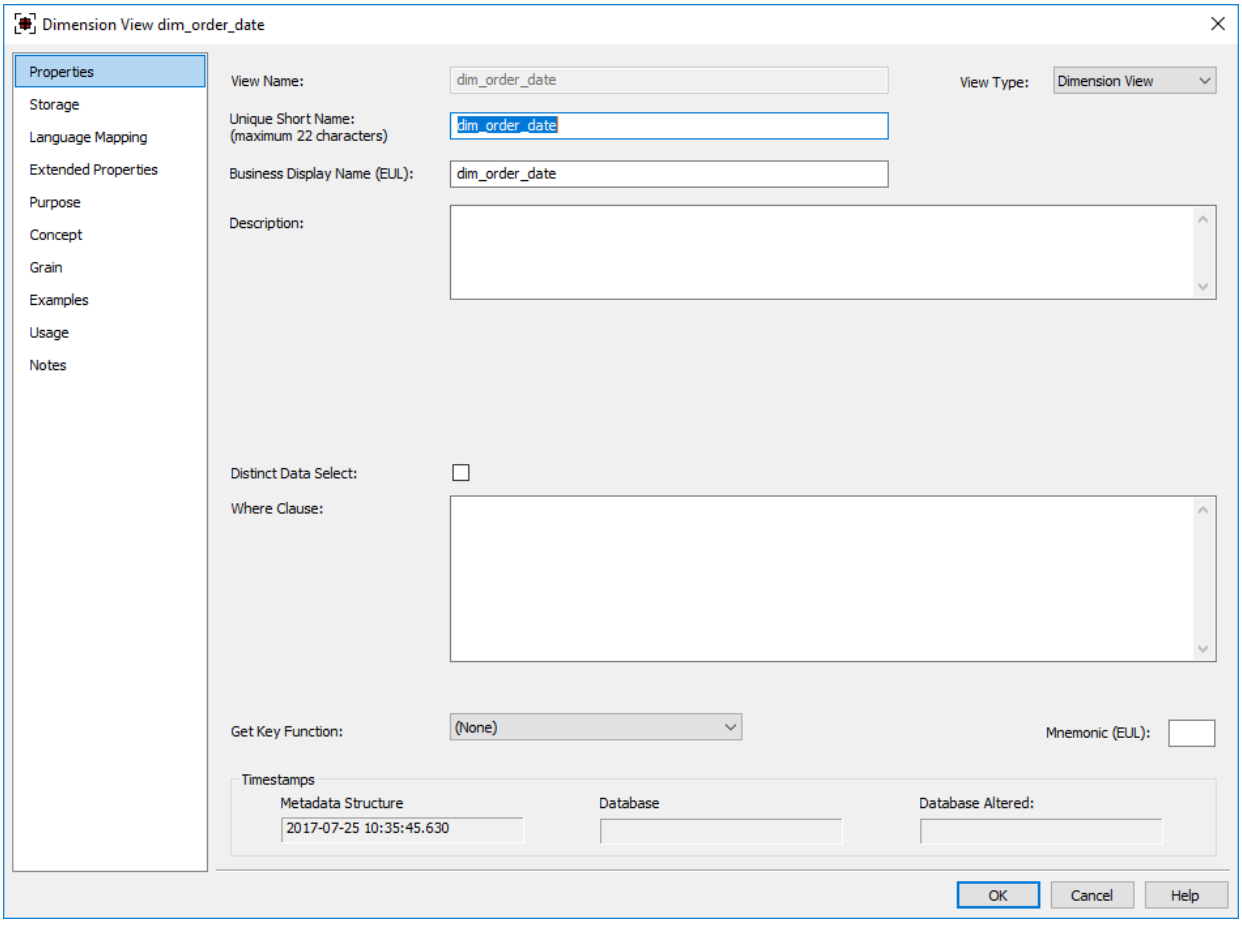


© Copyright, WhereScape Software Ltd All rights reserved.

4. The **View Column Definition** window appears to provide a means of re-mapping some of the column names in the view if required.

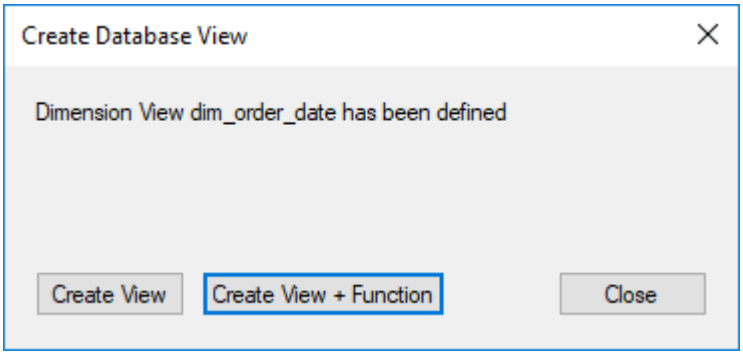


5. The View properties window appears:
   - Select the **Distinct Data Select** check box if you want the view to return only different values.
   - A **'Where'** clause could be entered, but this can be done later. Normally you would just click **OK**.

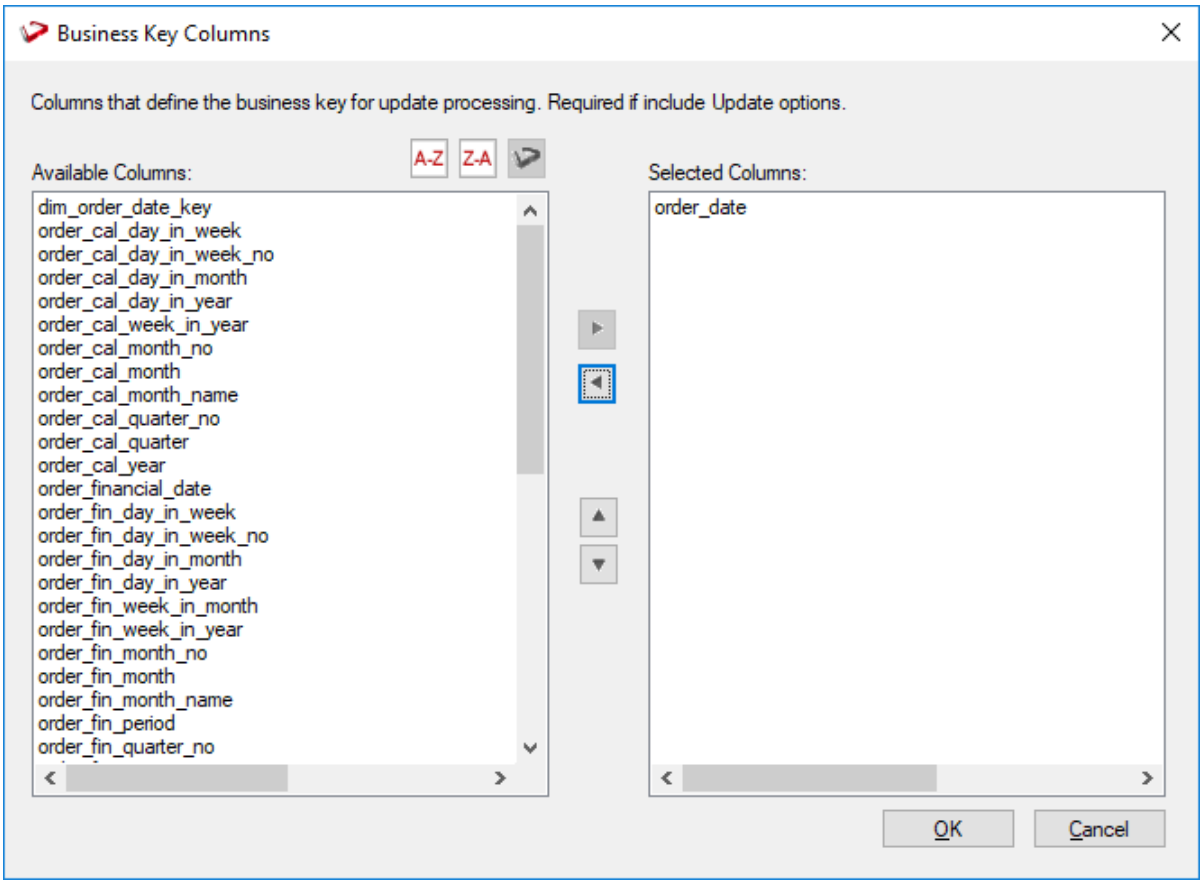


© Copyright, WhereScape Software Ltd All rights reserved.

6. A prompt appears to indicate that the view has been defined.
   - Because this is a Dimension view, you are given the option to create both the view and an associated Get Key function. For any other view type, a Get Key function is not created.



7. If you click **Create View + Function**, the following window appears which enables you to select a business (natural) key for the Dimension view.



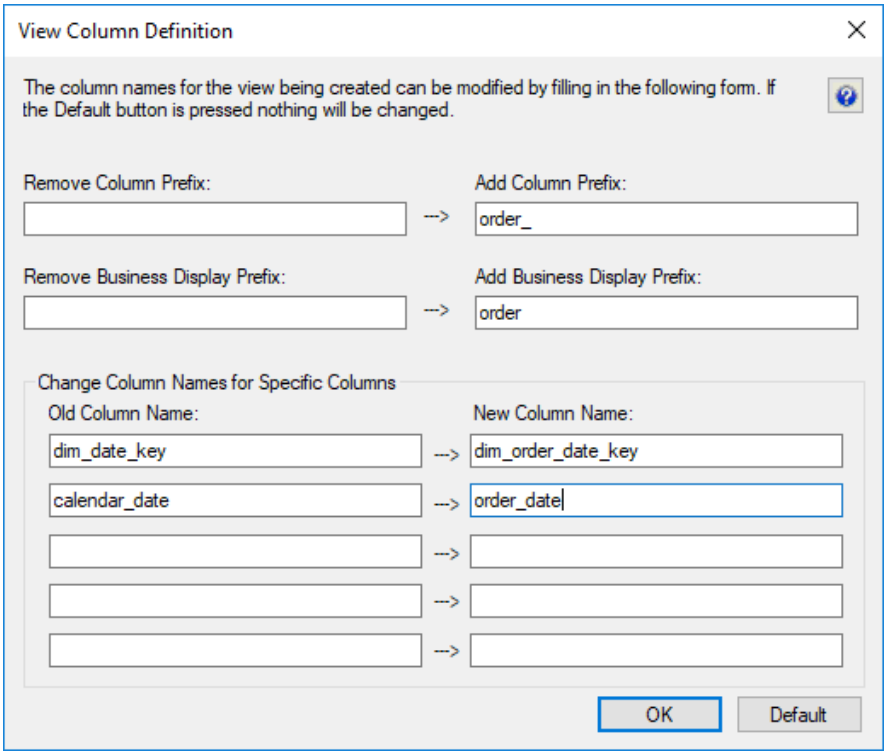
© Copyright, WhereScape Software Ltd All rights reserved.

View Column Re-mapping

The **View Column Definition** window is used for the automated re-mapping of certain column names. It provides an easy method for changing the column names for a large number of columns, when creating a View.

The various actions undertaken as a result of entries in this window can all be done or reversed manually by changing the individual column properties.

The various fields are described below:



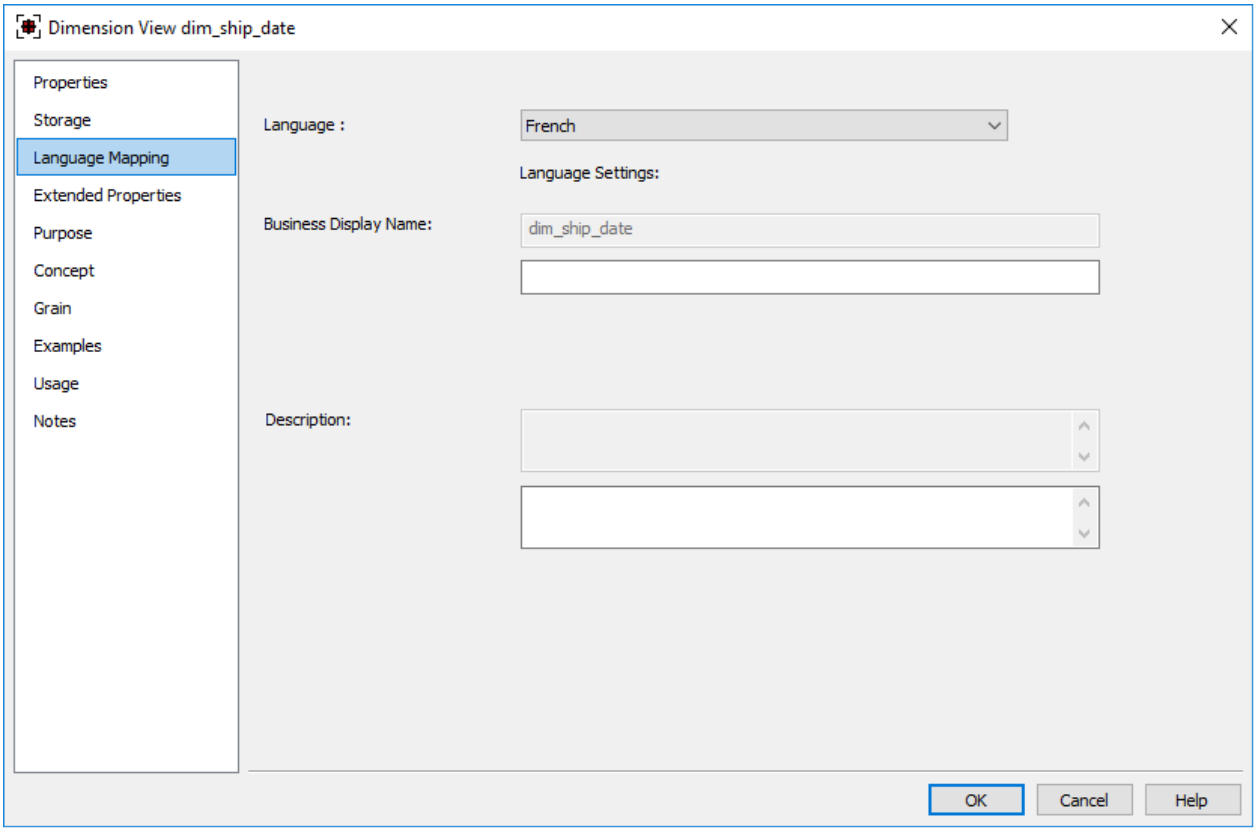
Fields	Description
Remove column prefix	If the columns in the source table are prefixed then that prefix can be removed by entering it in this field. An example may be a date dimension which has each column prefixed with date_ (e.g. date_month, date_year etc.). If this field is left blank then no removal action is taken.
Add column prefix	If required a prefix can be added to each column name. This is particularly useful when defining a date dimension view where you would like each column to be prefixed by the date type.
Remove business display prefix	As per the column names, it may be required to remove a prefix from the business display fields. If so enter the prefix to remove in this column.
Add business display prefix	The business display fields are used in the creation of the glossary. It is therefore quite useful to prefix these display fields with an identifier for the view being created. It is assumed that these business display names will be carried forward to the end user layer. Enter a value in this field to prefix the business display name fields for each column. It is normal to include a space at the end of this field.
Old column name	Up to five individual column names can be re mapped. Enter the column name as it appears in the source table in one of the **Old Column Name** fields in order to re map that column name. The business display name is also changed to match.
New column name	Place a new column name alongside any existing column name you wish to re map. In the example dialog above a column named **calendar_date** is being renamed to **order_date** in the view.

© Copyright, WhereScape Software Ltd All rights reserved.

Dimension View Language Mapping

The **Dimension View Properties** screen has a **Language Mapping** tab.

1. Select the language from the drop-down list and then enter the translations for the **Business Display Name** and the **Description** in the chosen language.
2. The translations for these fields can then be pushed through into OLAP cubes.



Tip

Languages can be added and maintained from the **Tools > Language Options** menu. Refer to **Language Options** for details.

Fields	Description
Business Display Name	Translated Value of Business Display Name.
Description	Translated Value of Description.

© Copyright, WhereScape Software Ltd All rights reserved.

Non-Dimension Object Views

WhereScape RED also supports creating views of objects other than dimensions (Load tables, Facts, Stage tables, Aggregates, etc.).

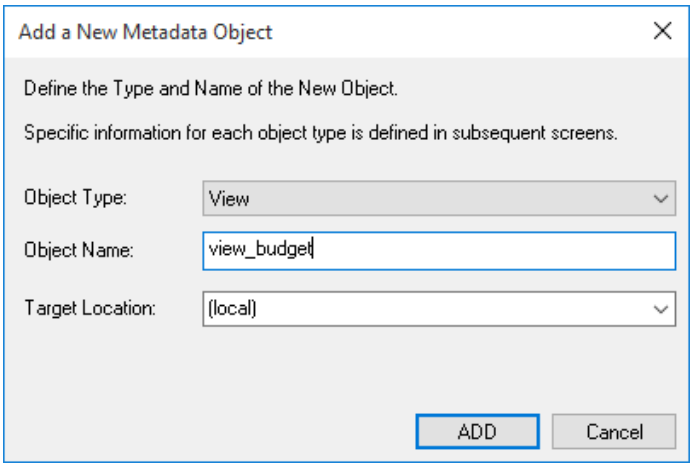
A Fact View is a database view of a Fact table. It may be a full, partial view, or a view incorporating both the fact data and some dimensional attributes.

It is typically used in cases where a subset of information is to be provided for specific end user communities. It is common that the 'Where' clause be populated in fact views, either to join the dimension tables or restrict the data that the view represents.

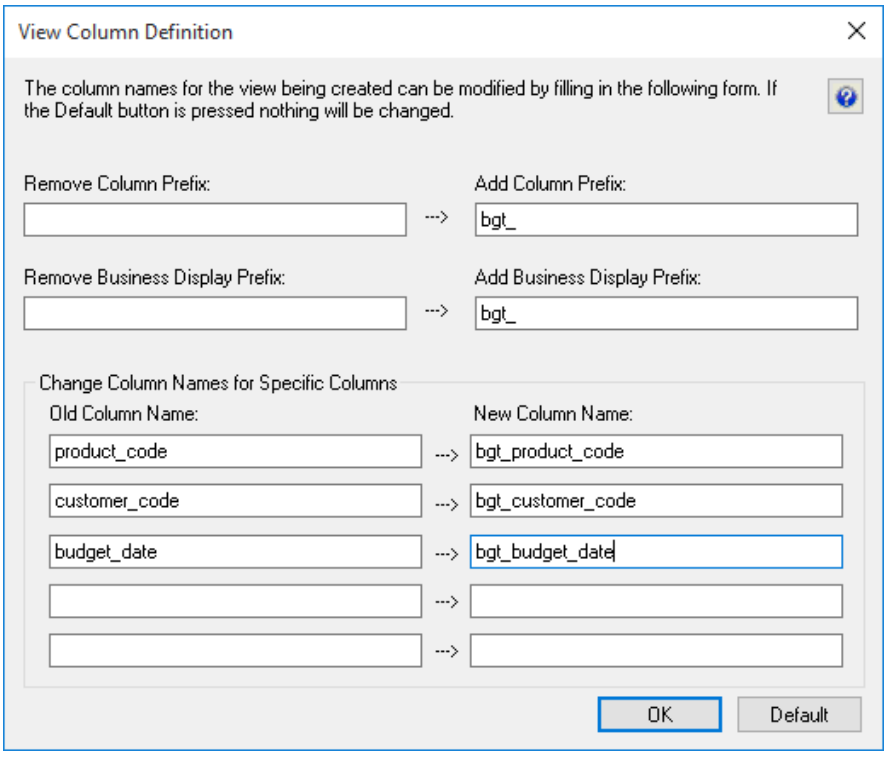
Similarly, views may be created over Staging tables or any other **table** object.

The creation of non-dimension table views is the same process as that for Dimension views:

1. Drag a table from the right pane into the middle pane.
   - The **Add New Metadata Object** window defaults the object type to a **View**.
   - Change the View name as required, and click **Add**.

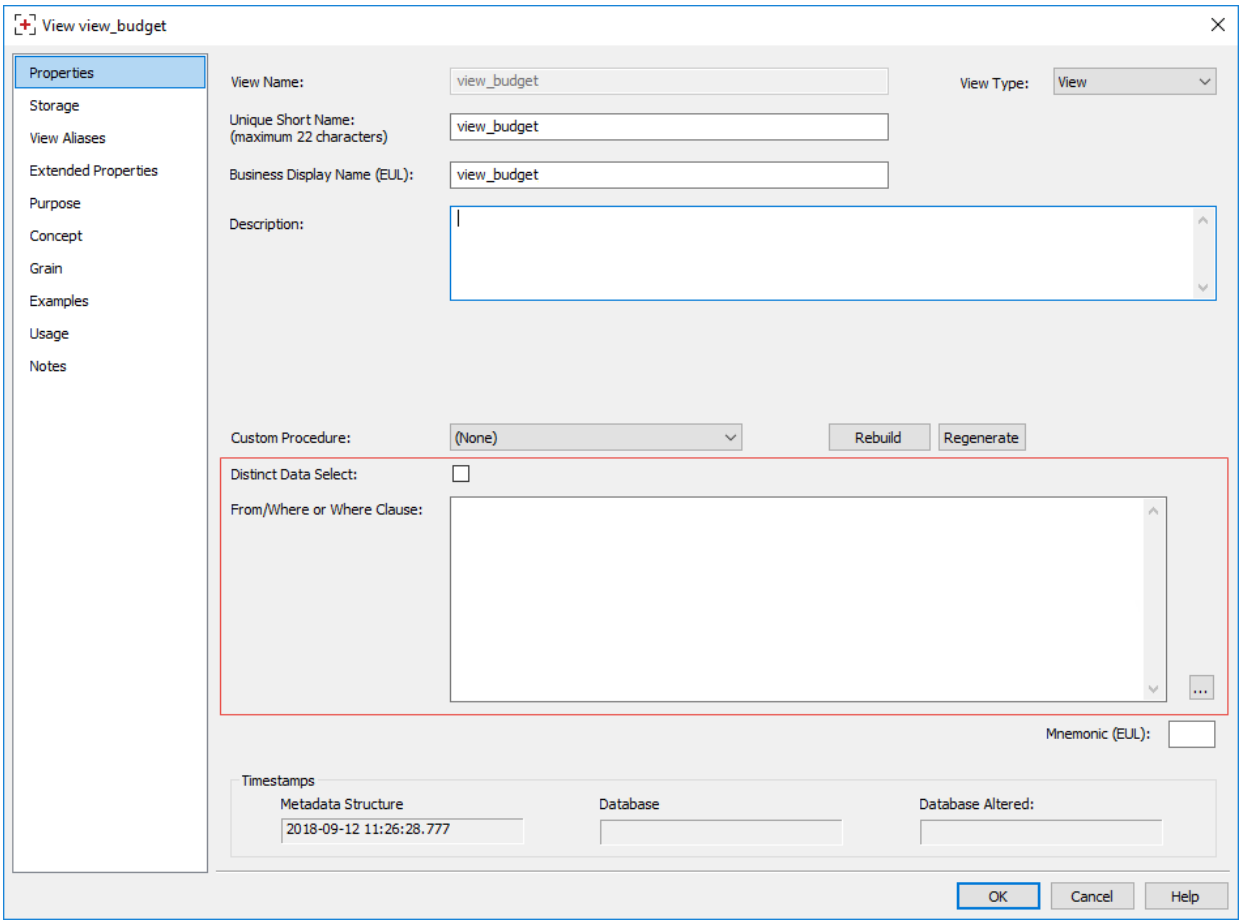


2. The **View Column Definition** window enables the automated re-mapping of certain column names. It provides an easy method for changing the column names for a large number of columns when creating a view.



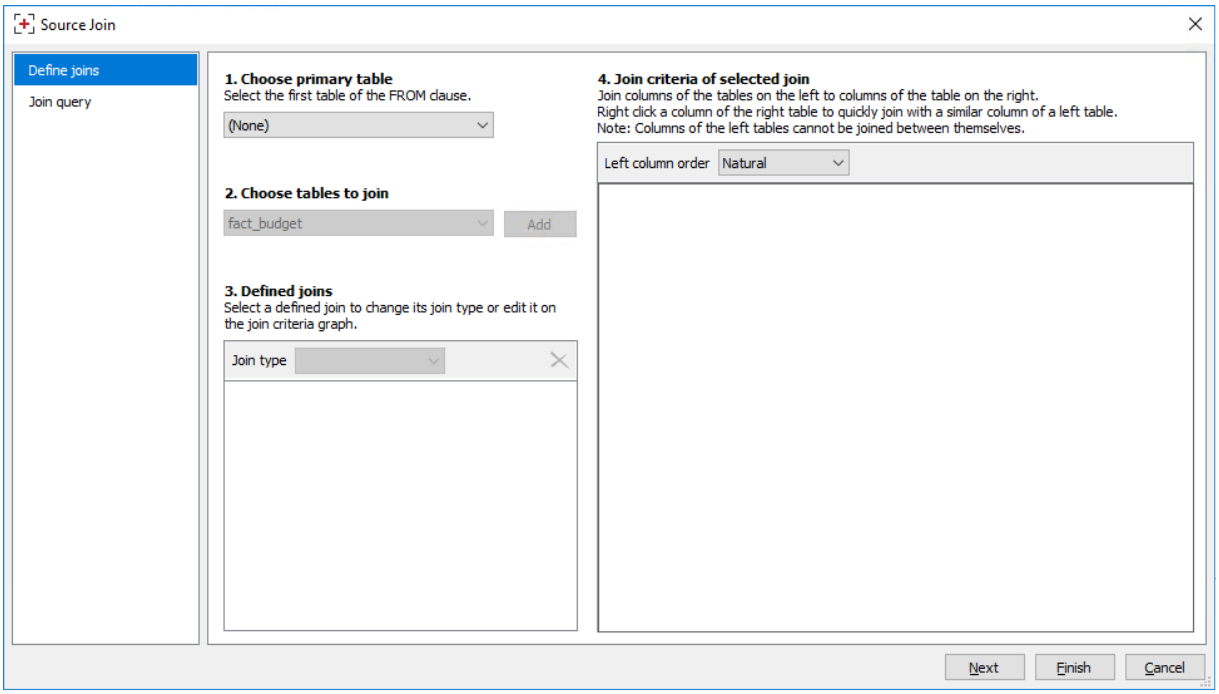
© Copyright, WhereScape Software Ltd All rights reserved.

3. The **View** properties window displays:



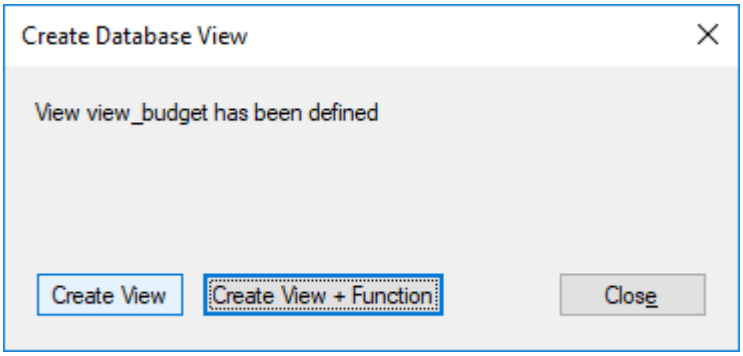
Change the following properties, if required:

- Select the **Distinct Data Select** check box if you want the view to return only distinct values.
- A **From/Where** clause could be entered but this can also be done later.
  Clicking the ellipses button opens the **Source Join** wizard shown below. Refer to **Generating the Dimension Update Procedure > Source tab > Joining multiple source tables** for details.

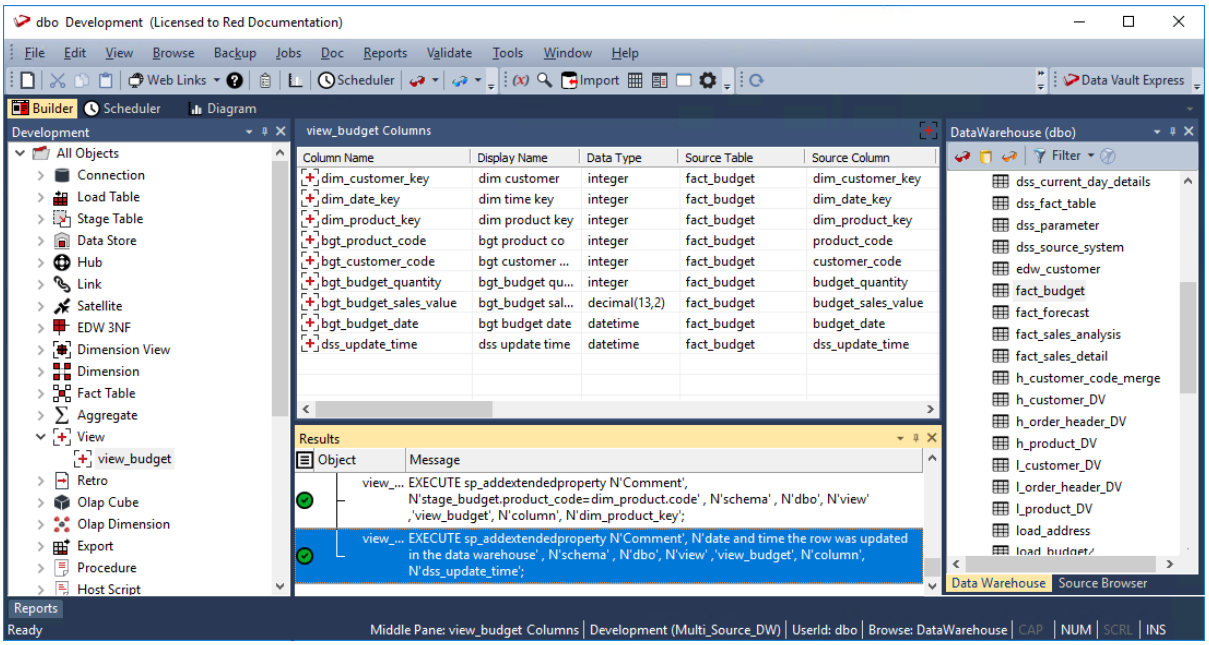


© Copyright, WhereScape Software Ltd All rights reserved.

4. Click **OK** in the **View** properties window, after completing view definition. A prompt is displayed to indicate that the view has been defined, click **Create View**.



The columns of the **View** table is displayed in the middle pane.



Note

If dimension attributes are being added (e.g.: to a fact view) then drag these attributes into the views column list, create the view and edit the 'Where' clause for the view.

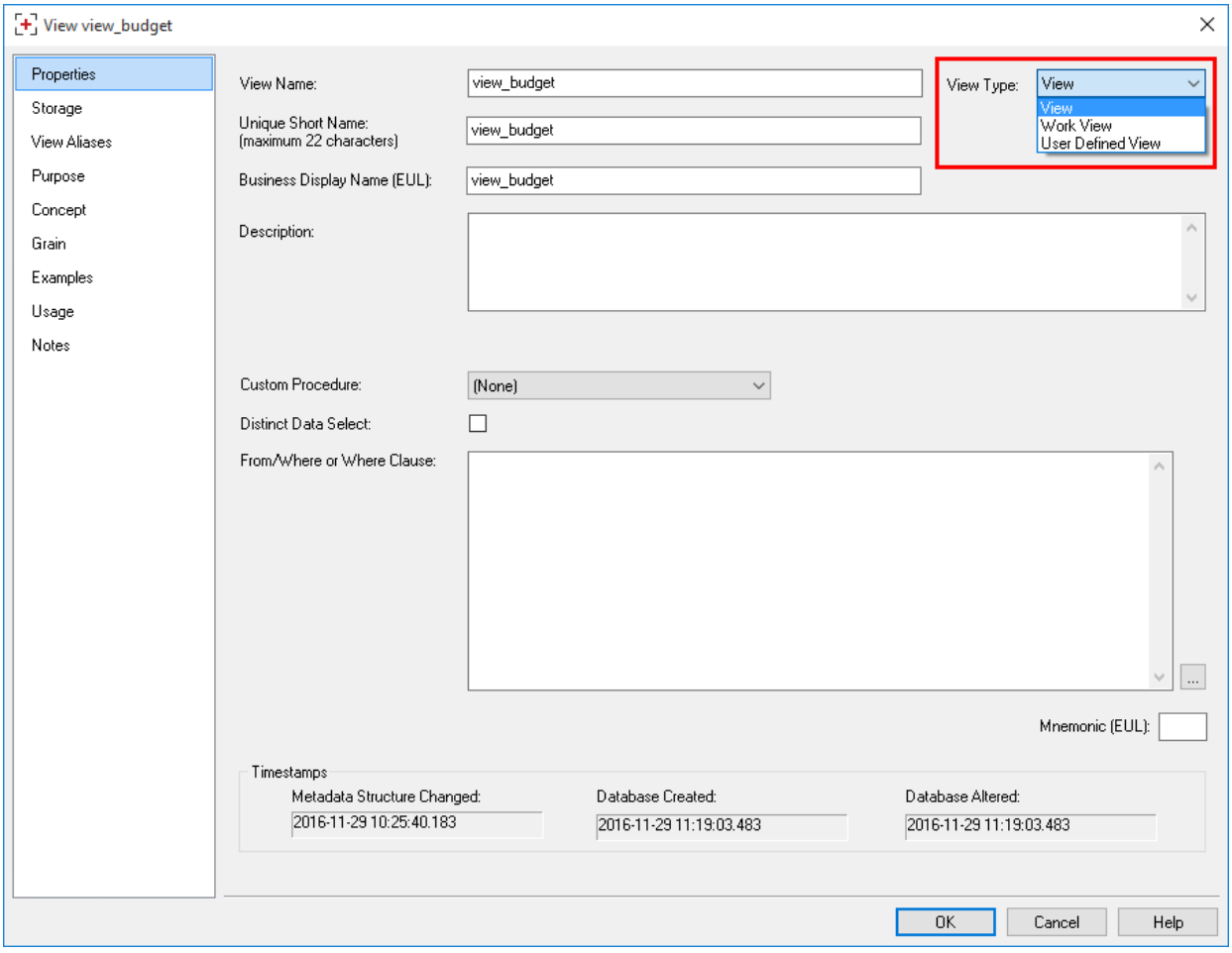
© Copyright, WhereScape Software Ltd All rights reserved.

Creating a Custom View

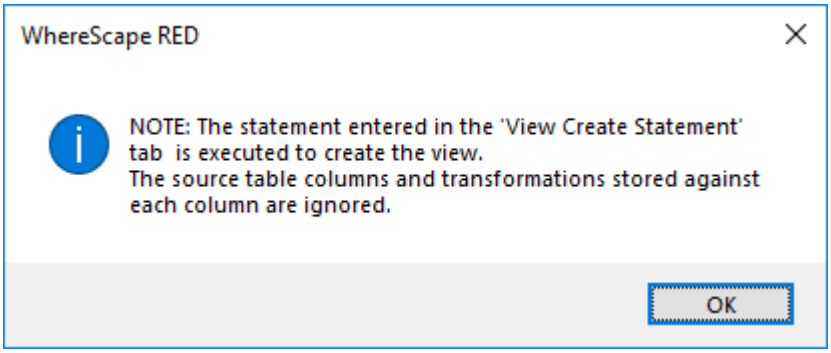
A custom view can be created within WhereScape RED to handle views that are not strictly one to one such as where multiple tables are joined or where a complex condition is placed on the view. There are two options for custom views, the first where the columns are defined in WhereScape RED and the 'Select' component of the view is customized. The second option is where the view is totally customized and no columns need to be defined in WhereScape RED, although it is good practice to still define the columns for documentation purposes.

Creating a Custom or 'User Defined' view:

1. Create a View either by dragging a table in or adding a new object.
2. Change the table type to **User Defined View** from the **View Type** drop-down list.

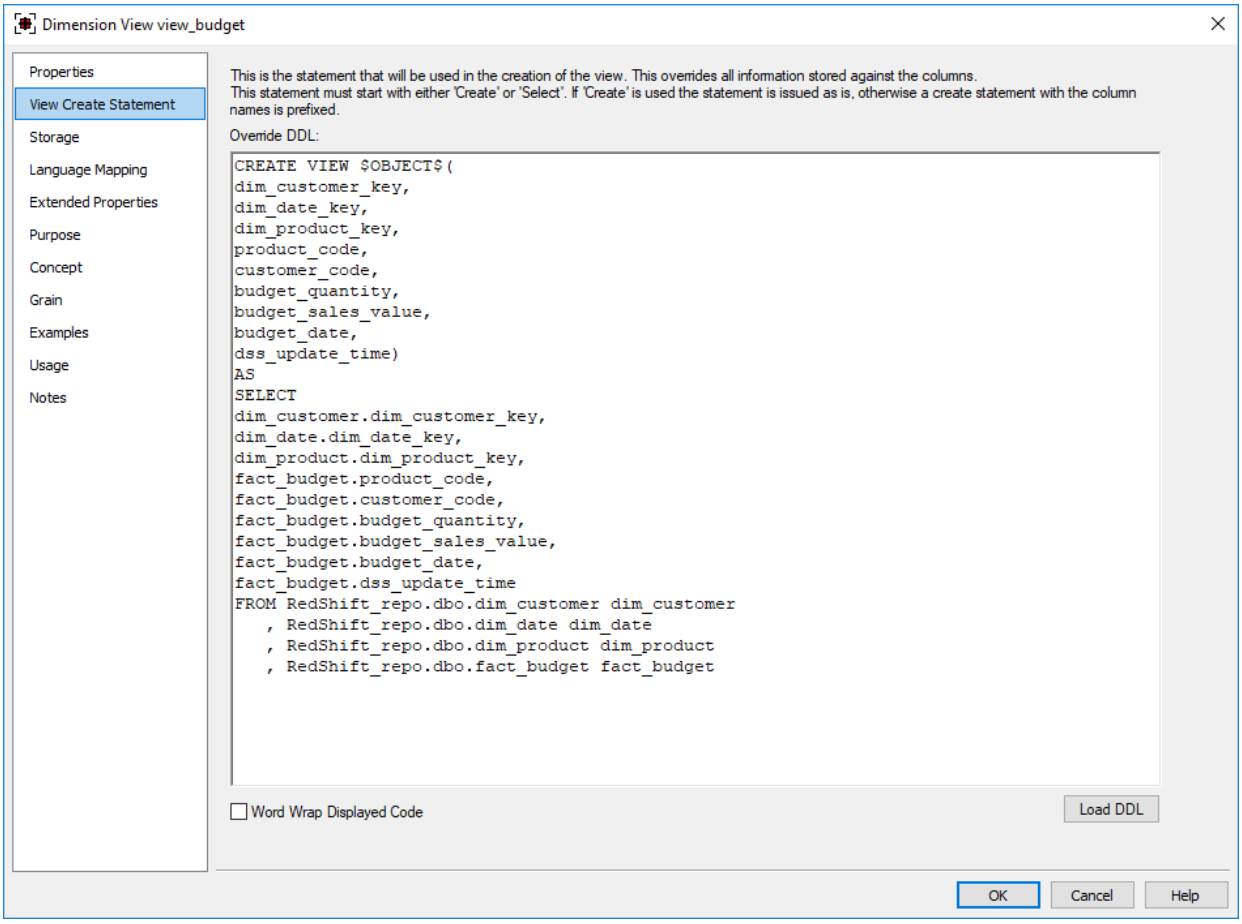


3. The following message is displayed. Click **OK**.



© Copyright, WhereScape Software Ltd All rights reserved.

4. Edit the new tab **View Create Statement** and insert the SQL Statement that will be used to create the view.
    - This SQL Statement must start with either **Create** or **Select**.
    - If **Create** is used then the columns in the view are ignored and the statement will be issued to create the view.
    - If the statement starts with **Select** then WhereScape RED will build up a view create statement from the column names and the supplied **Select** clause.
    - There is also a **Load DDL** button on the bottom right corner to get a sample **Select** statement, based on the columns in the view and any transformations.



© Copyright, WhereScape Software Ltd All rights reserved.

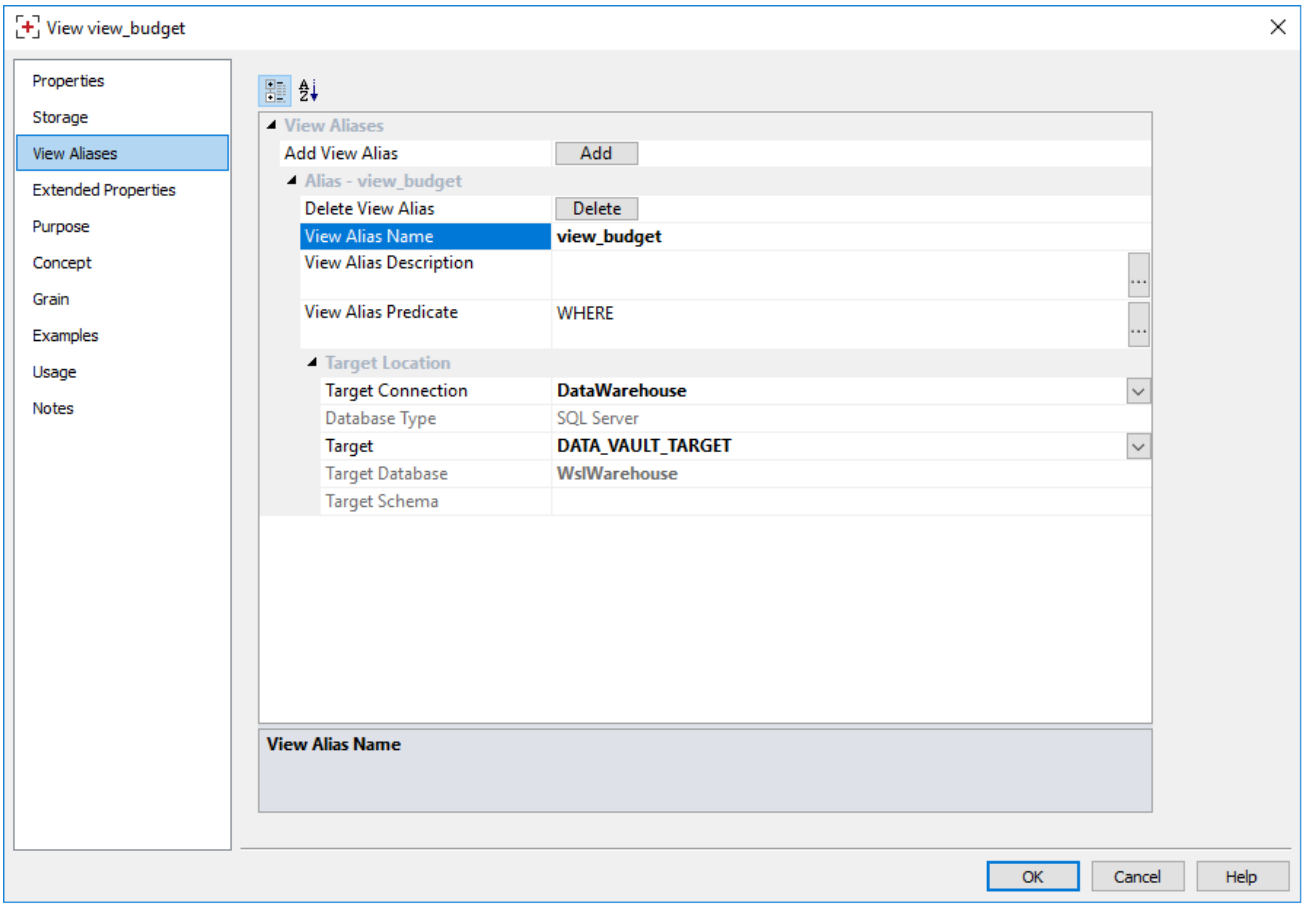
View Aliases

A View Alias provides the ability to create security views on SQL Server in an alternate schema.

Note

View Aliases are intended as a presentation object so that views over BI views could either mask a subset of the available data and/or have independent views in different databases/schema (SQL Server and Oracle). Because they are not stored as standard RED objects, they cannot be used as source tables for downstream processing.

The **View Aliases** tab enables you to define additional/replica views.



Fields	Description
Add button	Enables you to add a View Alias which adds a new copy of the view. This enables you to define new View Alias which is an alternate definition that will ultimately exist as another database view with a different name, predicate and/or target location.
Delete button	Enables you to delete a View Alias and remove its metadata details after closing the screen.
View Alias Name	The view alias name that is used as the alternate database view name.
View Alias Description	Description of the view alias.
View Alias Predicate	Optional 'Where' clause to include in the alternate view definition.
Target Connection	The name of the WhereScape RED connection that identifies the target location to create the alternate view.
Database Type	[Read Only] The database type of the selected target connection.
Target	The target that defines the database and schema for the table. Leave blank for the

© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
	connection default.
Target Database	[Read Only] Target database identifier (e.g. Oracle SID or TNS Name, Teradata TDPID) or Database Name (e.g. as in DB2 or SQL Server) to reference the alternate view.
Target Schema	[Read Only] The target Database/User schema for the alternate view. Leave blank to use the default schema,

© Copyright, WhereScape Software Ltd All rights reserved.

Analysis Services OLAP Cubes/Tabular Models

OLAP Cubes

A **cube** is a set of related measures and dimensions that is used to analyze data.

- A **measure** is a transactional value or measurement that a user may want to aggregate. The source of measures are usually columns in one or more source tables. Measures are grouped into **measure groups**.
- A **dimension** is a group of attributes that represent an area of interest related to the measures in the cube and which are used to analyze the measures in the cube. For example, a customer dimension might include the attributes:
    - Customer Name
    - Customer Gender
    - Customer City

These enable measures in the cube to be analyzed by Customer Name, Customer Gender, and Customer City. The source of attributes are usually columns in one or more source tables. The attributes within each dimension can be organized into hierarchies to provide paths for analysis.

A cube is then augmented with **calculations**, key performance indicators (generally known as **KPIs**), **actions**, **partitions**, **perspectives**, and **translations**.

The information required to build and support an Analysis Services cube and its surrounding structure is reasonably complex and diverse. In attempting to automate the building of Analysis Services cubes, WhereScape RED has simplified and restricted many of the functions available to the cube designer. WhereScape RED includes most of the commonly used capabilities and the components that logically fit into the methodology incorporated within WhereScape RED.

WhereScape RED broadly provides functionality to manage all of the above, except for perspectives and translations. These can be created outside of WhereScape RED, scripted in xmla and executed from within WhereScape RED. Features of cubes that are not supported in WhereScape RED can be added to the cube via the Microsoft tools. These altered cubes can still be processed through the WhereScape RED scheduler, and the cube should be documented within WhereScape RED to explain the post creation phases required.

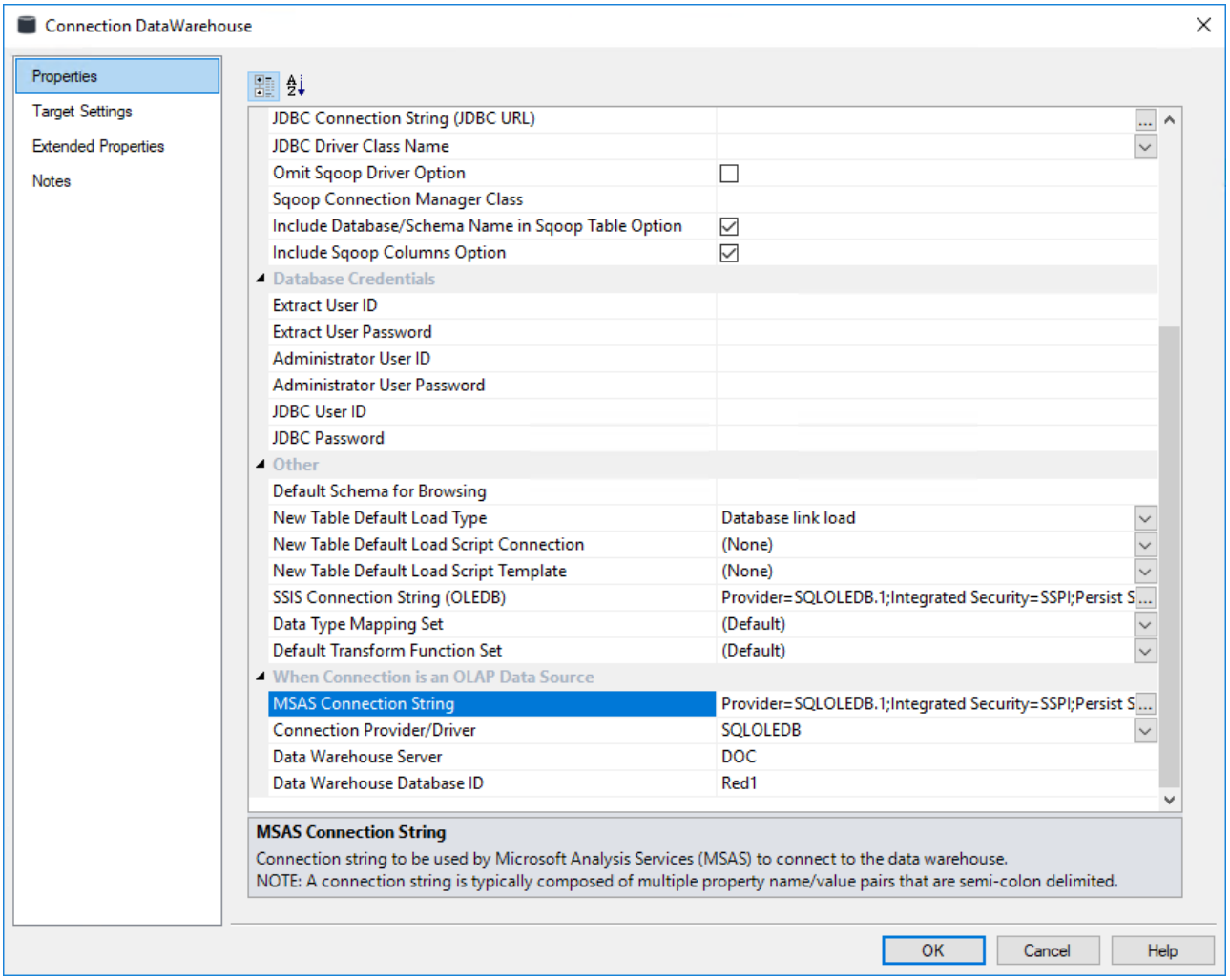
As a general rule, once a cube or a component of a cube is created on the Analysis Services server it cannot be altered through WhereScape RED. The OLAP object can be dropped and recreated easily using RED. New OLAP objects defined in RED (e.g. additional calculations or measures) can be added by recreating the cube.

© Copyright, WhereScape Software Ltd All rights reserved.

Defining the Data Source for the OLAP Cube

Before we can create an OLAP cube, we first need to set up the data warehouse to be used as a source for Analysis Services cubes.

In the **Data Warehouse** Properties screen, the fields in the section **When Connection is an OLAP Data Source** are required.

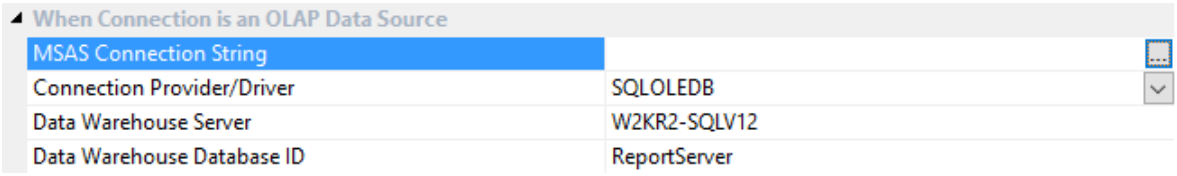


Fields	Description
MSAS Connection String	Connection string to be used by Microsoft Analysis Services (MSAS) to connect to the data warehouse. **Note:** A connection string is typically composed of multiple property name/value pairs that are semi-colon delimited.
Connection Provider/Driver	Name of the Connection Provider/Driver to use to connect to the data warehouse database when it is used as the data source for OLAP cubes. Set to **SQLOLEDB** for SQL Server, **TDOLEDB** for Teradata.
Data Warehouse Server	Data Warehouse Server Name, which is used when the data warehouse is used as the data source for OLAP cubes. Set this to the TDPID for Teradata.
Data Warehouse Database ID	Data Warehouse Database Identifier (e.g. Oracle SID or TNS Name, Teradata TDPID) or Database Name (e.g. as in DB2 or SQL Server), which is used when the data warehouse is used as the data source for OLAP cubes.

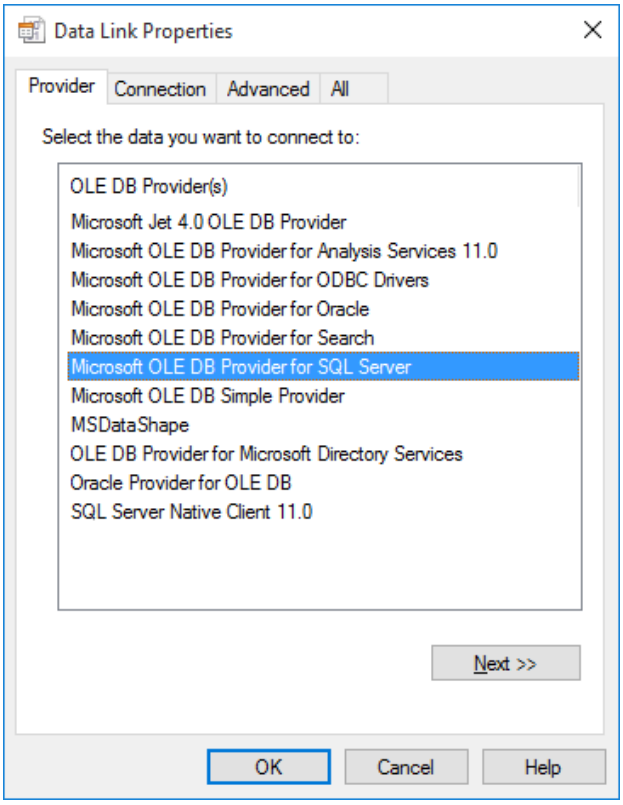
© Copyright, WhereScape Software Ltd All rights reserved.

Building the OLAP Connection String

1. The **OLAP Connection String** is built using a wizard. To activate the wizard, click the expander button.

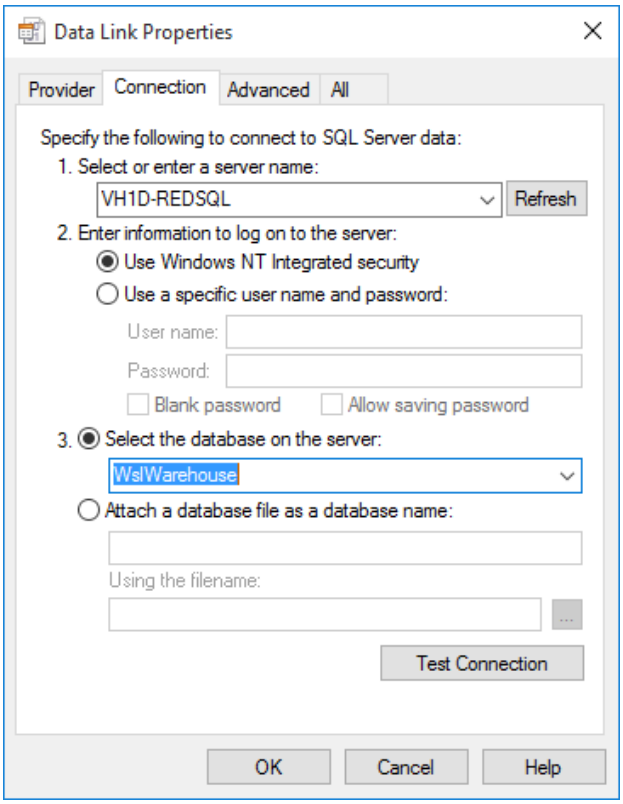


2. In the **Provider** tab, select the **OLE DB Provider** and click **Next**.

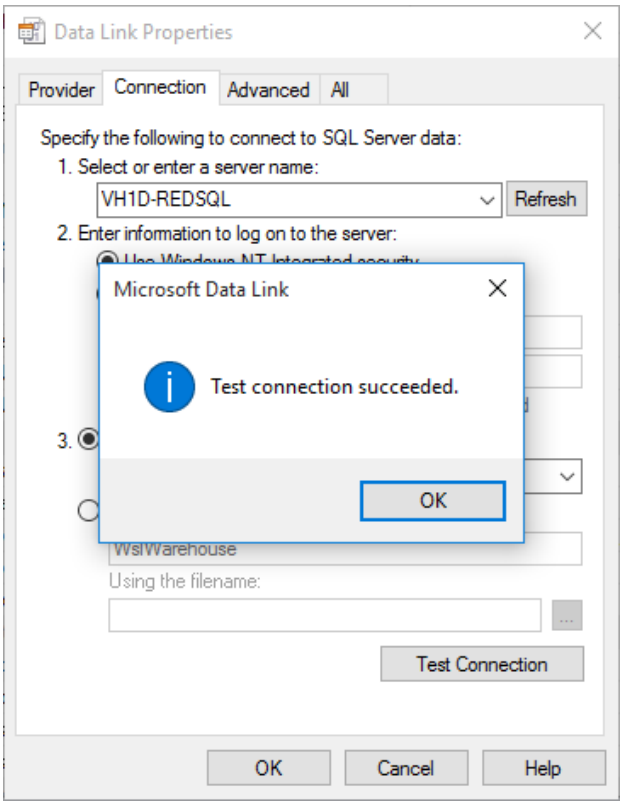


3. Enter the connection details and click **Test Connection**.

© Copyright, WhereScape Software Ltd All rights reserved.



4. Click **OK** in the success message and then **OK** again on the **Data Link Properties** screen.



5. The OLAP connection string is displayed.

© Copyright, WhereScape Software Ltd All rights reserved.

OLAP Defining an OLAP Cube

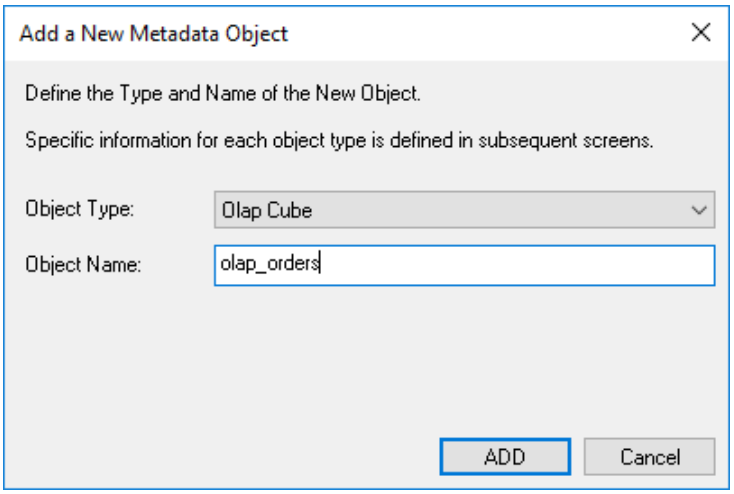
OLAP Cubes can be created from fact, fact view or aggregate objects. A single cube can contain dates from multiple source star schemas, each defined with a measure group. An OLAP Cube consists of many parts namely, measure groups, measures, calculations, actions, dimensions, dimension hierarchies, dimension attributes and dimension attribute relationships.

It is strongly recommended that drag and drop is used to create an OLAP Cube in order that all the components are set up correctly. OLAP Cubes can utilize a hierarchical structure in the dimensions to facilitate user queries. Therefore, each dimension present in an OLAP Cube must have either a hierarchy of levels or attributes and relationships. The hierarchies are defined against the underlying dimensional attributes which can be inherited from the source dimension metadata. Individual attributes can be added to the dimension after the OLAP Cube or OLAP Dimension metadata has been created.

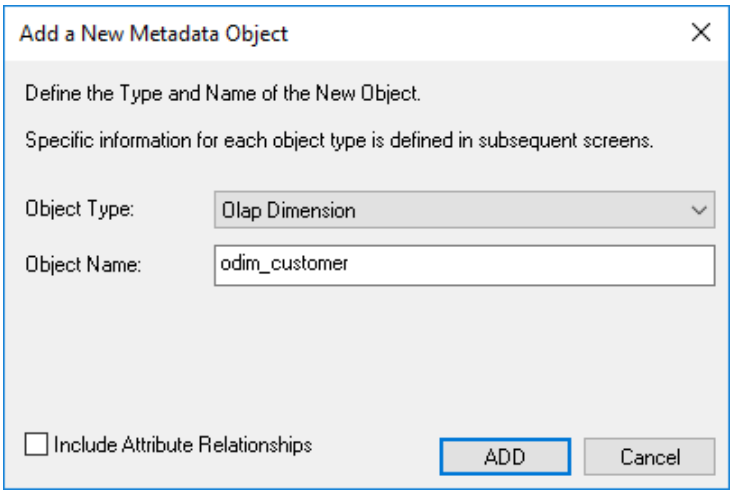
Building a New OLAP Cube

To create an OLAP Cube proceed as follows:

1. Double-click on the **OLAP Cube object group** to make the middle pane a cube drop target.
2. Select the **data warehouse connection** to browse in the source pane. The connection can be selected by right-clicking the Data Warehouse connection in the Object list pane and choosing **Browse Source System**.
3. Drag a **Fact** table from the source pane into the target pane.
4. Set the cube name in the **Add a New Metadata Object** window and then click **ADD**.



5. A prompt is displayed for any OLAP Dimensions that do not already exist that are required for this cube (based on the fact table metadata). Set the dimension name in the **Add a New Metadata Object** window and then click **ADD**.

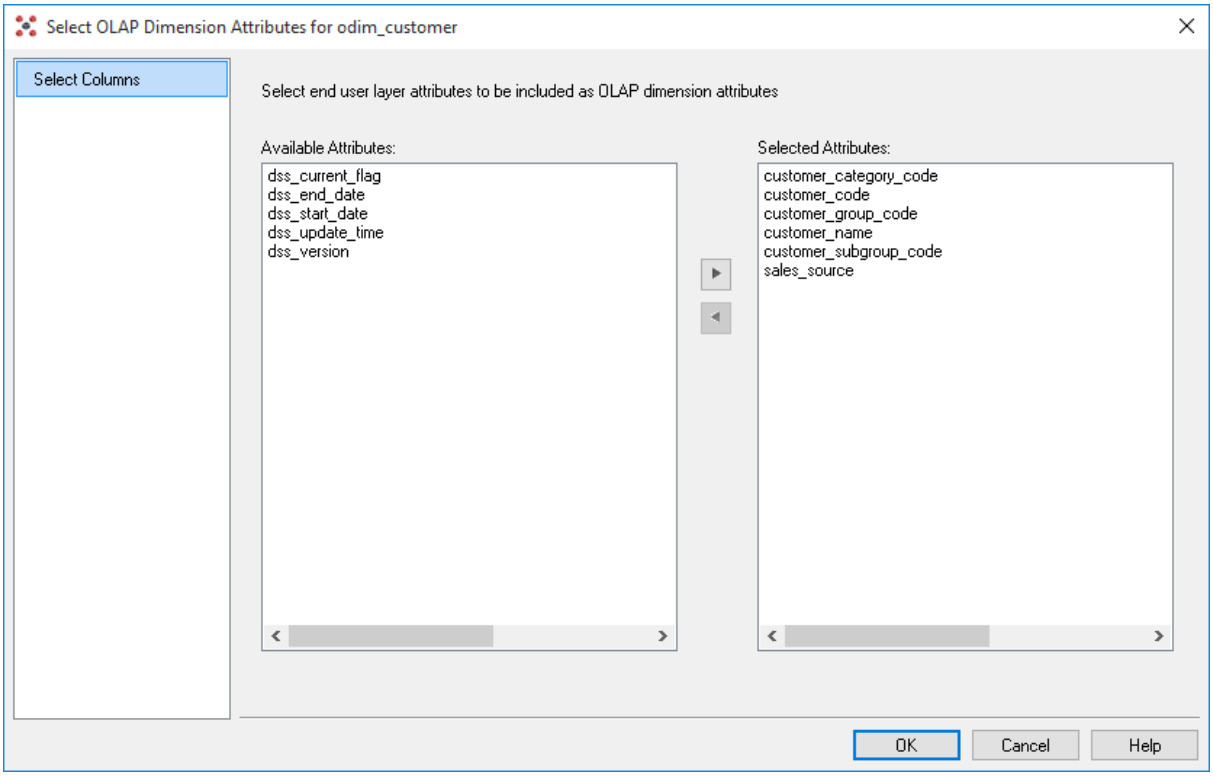


© Copyright, WhereScape Software Ltd All rights reserved.

Note:

If you wish to include Attribute Relationships in Analysis Services for this dimension, click the **Include Attribute relationships** check box.

6. The following window appears, prompting you to select the attributes to be included in the **Customer** OLAP dimension.



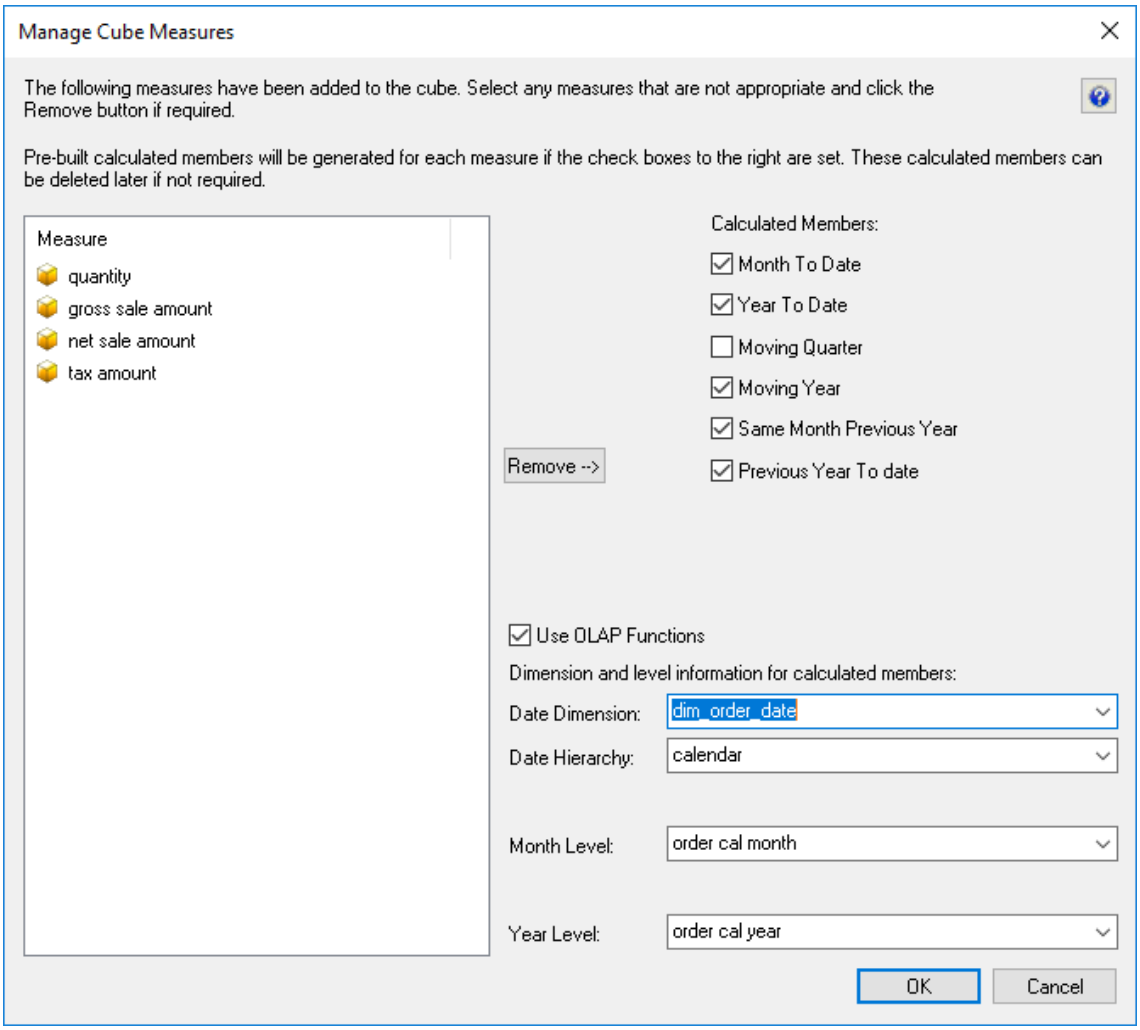
The attributes available for selection are in the left column. To select an attribute, click the attribute in the left column and click **>**. This moves the attribute to the right column.

To de-select an attribute, click the attribute in the right column and then click **<**. This moves the attribute to the left column.

Repeat Step 5 for each dimension as required.

7. A window appears with a list of the fields that are to be added as measures. Remove any columns that are not measures. A measure is a column that uses the sum, count, min or max of the column. Calculations can be chosen if required at this point. A date dimension must be present along with a hierarchy to enable the definition of these calculated members.

© Copyright, WhereScape Software Ltd All rights reserved.



8. During cube creation, the **Adding Cube Measures** window is displayed. In this window the following options are provided:
   - Measure, provides a list of measures that can be aggregated (e.g. using Sum, Count, Min, Max or Distinct Count). By default, WhereScape RED shows all attributes in the Fact table that are defined as numeric and additive. Attributes that should not be considered measures can be removed using the **Remove** button.
   - **Calculated Member** options enable you to add some predefined date based calculated member definitions to be built against the cube. The standard calculations:
     - Month to date
     - Year to date
     - Moving Quarter
     - Moving Year
     - Same Month Previous Year
     - Previous Year to date

   These defines a calculated measure based on the associated drop-down boxes. There are two different ways that WhereScape RED implements these calculations which is dictated by the **Use OLAP functions** check box:
   - Using OLAP Functions - implements the calculations using MDX Expressions within the cube using date based MDX functions. These calculations are efficiently executed by Analysis Services.
   - Without using OLAP functions - implements the calculations using an MDX Filter function built over date dimension attributes. This option leverages the flags from the relational date dimension and ensures that a query using the calculations in the Cube matches an equivalent query against the star schema and is particularly useful if non-standard date periods are used.
9. The cube and dimensions is created in WhereScape RED metadata and the cube measures is displayed.

© Copyright, WhereScape Software Ltd All rights reserved.

Setting Cube Properties

The properties of the cube must be completed before we can create the cube in the Analysis services database. Most of the elements in the properties screen has default values, but each of the following columns probably need to be completed.

1. The **Connection** to the Analysis services server must be defined within the cube properties. This connection is a connection object. If no such connection exists then a new connection of type must be created and configured. SQL Server 2005 or 2008 Analysis Services use a connection type of "Analysis Server 2005+". This connection name must then be chosen in the cubes properties.
2. A **Cube Database Name** must be selected. A new database name can be created by selecting **(Define New Cube Database...)** from the drop-down list. This database name is the database that the cubes will reside on the Analysis services server.
3. The **Data Source Connection** must be defined and the three derived values shown under this connection must be present. If there is nothing in the three fields below the data source connection then the connection object will need to be modified. The provider type is normally MSDAORA. if the data warehouse is an Oracle database or SQLOLEDB if the data warehouse is SQL Server. The Server is either the SQL Server database server, or the Oracle TNS name. The database is the data warehouse database name.

OLAP Inspecting and Modifying Advanced Cube Properties

After the basic OLAP Cube has been defined, various properties of the OLAP Cube can be inspected or modified:

Properties	Description
Measure Groups	1. Display the Measure Groups by right-clicking the cube name and selecting **Display Measure Groups**. 2. Change the Measure Group properties by right-clicking the measure group and selecting **Properties**.
Measures	1. Display all the Measures associated with a cube by right-clicking the cube name and selecting **Display Measures**. 2. Change the measure properties by right-clicking the measure name and selecting **Properties**.
Calculations	1. Display all the Calculated members defined the cube by right-clicking the cube name and selecting **Display Calculations**. 2. Change the Calculated members by right-clicking a calculation and selecting **Properties**.
KPIs	1. Display all the KPIs defined on the cube by right-clicking the cube name and selecting **Display KPIs**. 2. Change the KPIs by right-clicking the KPI name and selecting **Properties**.
Actions	1. Display all the Actions defined the cube by right-clicking on the cube name and selecting **Display Actions**. 2. Actions can be changed by right-clicking the Action name and selecting **Properties**.
Partitions	1. Display all the Partitions defined the Measure Groups that are associated with the cube by right-clicking the cube name and selecting **Display Partitions**. 2. Change Partitions by right-clicking the Partition name and selecting **Properties**.
Dimensions	1. Display all of the OLAP Dimensions associated with the cube by right-clicking on the cube name and selecting **Display Dimensions**. 2. Change the customizable OLAP Dimension properties by right-clicking the

© Copyright, WhereScape Software Ltd All rights reserved.

Properties	Description
	OLAP Dimension name and selecting **Properties**.
Measure Group Dimensions	1. Display the relationship of OLAP Dimensions to Measure Groups defined against the cube by right-clicking the cube name and selecting **Display Measure Group Dimensions**. 2. Change the customizable properties of the relationship of the OLAP Dimension to Measure Group by right-clicking the OLAP Dimension name and selecting **Properties**.

OLAP Creating an OLAP Cube on the Analysis Services Server

If all the tasks above are completed ,then it is now possible to create the cube on the Analysis Services server. Right-click the **OLAP Cube** name and then select **Create (Alter) Cube**.

WhereScape RED checks that key components of the cube are correct, before it proceeds to issue the create command.

The create cube menu option will perform the following tasks on the Analysis Services server:

- Create an Analysis Services database if the name specified is not already present.
- Create a Data Source with connection information back to the data warehouse based on the cube source information in the Data Warehouse connection.
- Create a Data Source View to support the cube objects defined
- Create the dimensions used by the cube as database shared dimensions if they do not already exist.
- Create the cube if it does not exist
- Create a partition for the cube.

OLAP Cube Objects

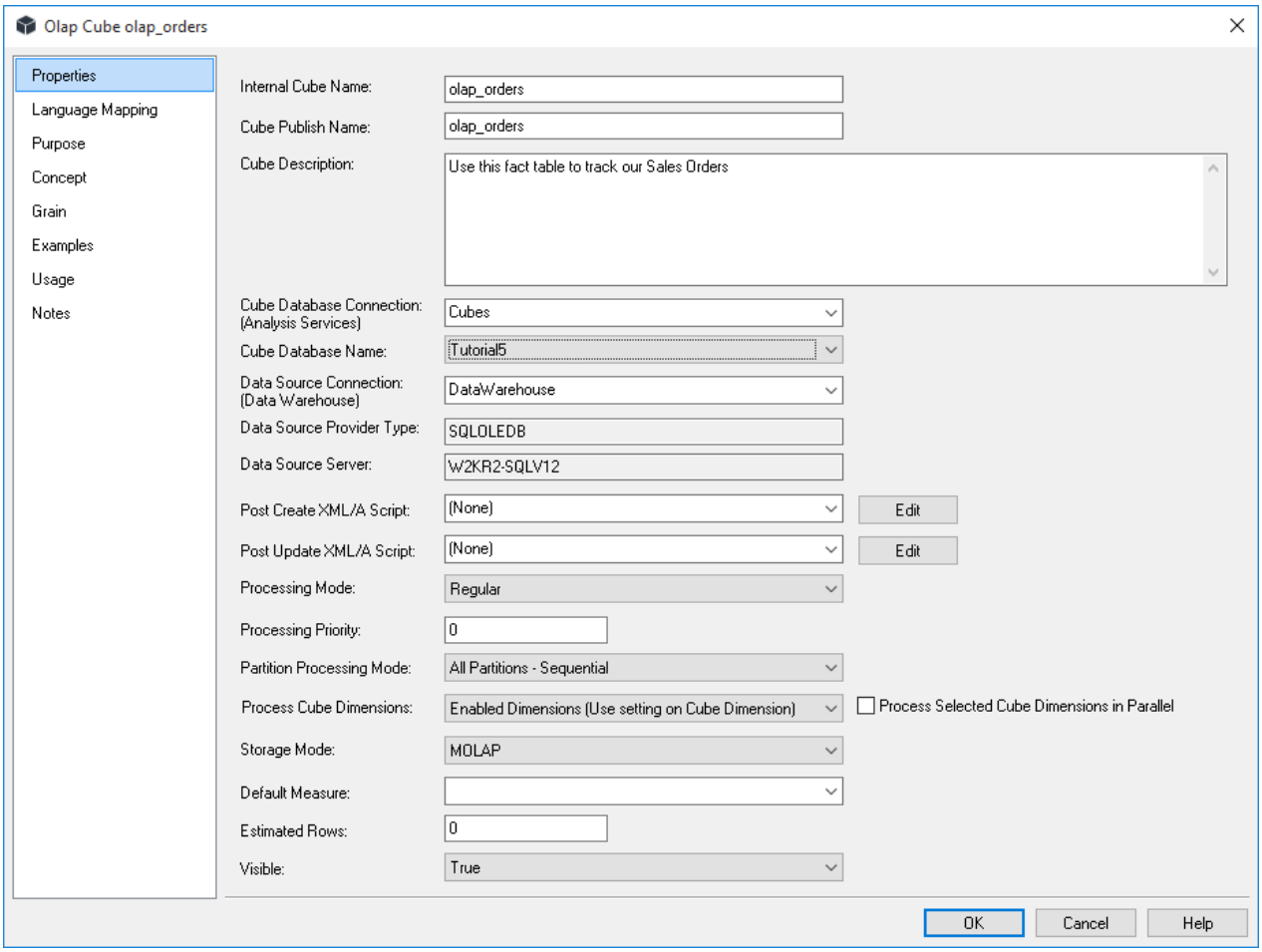
OLAP Cube Properties

The properties associated with a cube are described below. These properties relate both to the cube environment and to the cube itself.

There are seven tabs in the cube **Properties** screen. The first is the main properties, the second the processing and partitioning options and the rest are for documentation stored in the WhereScape RED metadata and displayed in the generated WhereScape RED documentation.

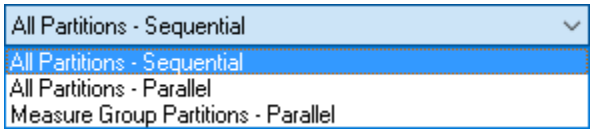
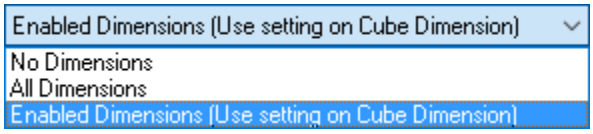
In order to see the cube properties, right-click the cube name and then select **Properties**.

© Copyright, WhereScape Software Ltd All rights reserved.



Fields	Description
Internal Cube Name	This is the name by which the cube is known within WhereScape RED. This name is limited to 64 characters in length and may contain only alphanumeric characters and underscores.
Cube Publish Name	This is the name of the cube in the Analysis Services server. It is not constrained in its content except for the limitations imposed on it by Analysis Services.
Cube Description	A description of the cube. This is used both in the WhereScape RED documentation and is also stored against the cube in Analysis Services.
Cube Database Connection	This field enables the selection of one of the existing connections defined within WhereScape RED. The connection must be of type 'Microsoft Analysis Server 2005+'. If no such connection exists then a new connection object must be created. This connection object is used to point to the Analysis Services server.
Cube Database Name	An Analysis Services server must have databases associated with it. Each database can contain one or more cubes, dimensions, data sources etc. Select the name of an existing database on the server from the drop-down list. To create a new Database name, select **(Define New Cube Database)** from the drop-down list and the displayed window enables you to register the name within the WhereScape RED metadata. Once registered, the name can then be selected as the database.
Data Source Connection	In Analysis Services the data source is the location of the data to be used to build the cube. It also defines the path for any drill through operations. This field provides a drop-down of the existing connections. The Data Warehouse connection must be chosen.
Data Source Provider Type	This field essentially defines what type of database the Data Warehouse is. The three options are:

© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
	• MSDAORA where the data source is an Oracle database • OLESQLDB where the data source is a SQL Server database, or • MSDASQL to define an ODBC source The MSDASQL provider type could be used for either Oracle or SQL Server data warehouses, but performance may be compromised. This field is Read-Only in the Properties screen. Its value is set in the properties of the data source connection.
Data Source Server	The data source server is also a read only field being sourced from the properties of the data source connection. For SQL Server, it defines the server on which the data warehouse database runs. For Oracle, it is the TNS names entry for the database.
Data Source Database	The data source database is also a read only field being sourced from the properties of the data source connection. For SQL Server, it defines the database in which the data warehouse runs. For Oracle, it is also the database name.
Post Create XML/A Script	This is an XML/A script that is run on the cube database when the cube is created. This script enables Analysis Services features to be added to the cube or cube database that have been built outside of WhereScape RED—for example security roles that have been defined for the cube can be recreated from the script, when the cube is created (or recreated).
Post Update XML/A Script	This is an XML/A script that is run on the cube database, when the cube is updated or processed via the scheduler. This script enables Analysis Services features to be added to the cube or cube database that have been built outside of WhereScape RED—for example security roles that have been defined for the cube can be recreated from the script when the cube is updated or processed.
Processing Mode	Gets or sets the index and aggregation settings for cube processing. The value indicates whether processed data is available after all required data aggregation has been completed (Regular) or immediately after the data has been loaded (Lazy Aggregations). This setting is used as the default for new measure groups and partitions created for the cube.
Processing Priority	Gets or sets the processing priority for the cube.
Partition Processing Mode	This option determines how partitions are updated, when a cube is updated.  • **All Partitions - Sequential** updates each cube partition sequentially. • **All Partitions - Parallel** all the cube partitions are updated in parallel. • **Measure Group Partitions - Parallel** processes each measure group sequentially with all partitions of that same measure group being updated in parallel.
Process Cube Dimensions	Determines whether to process the OLAP Dimensions related to the cube, as part of the processing of the cube. The options are to process the **enabled dimensions** only, to process **no dimensions**, or to process **all the dimensions**. Processing of specific Dimensions with the Cube can be enabled or deactivated via the **Process Cube Dimension With Cube** check box of each Dimension's Properties screen. Refer to **OLAP Cube Dimensions** for details. 
Process Selected Cube Dimensions in Parallel	Select this check box to enable all dimensions within the cube to be updated in parallel, instead of being updated sequentially.

© Copyright, WhereScape Software Ltd All rights reserved.

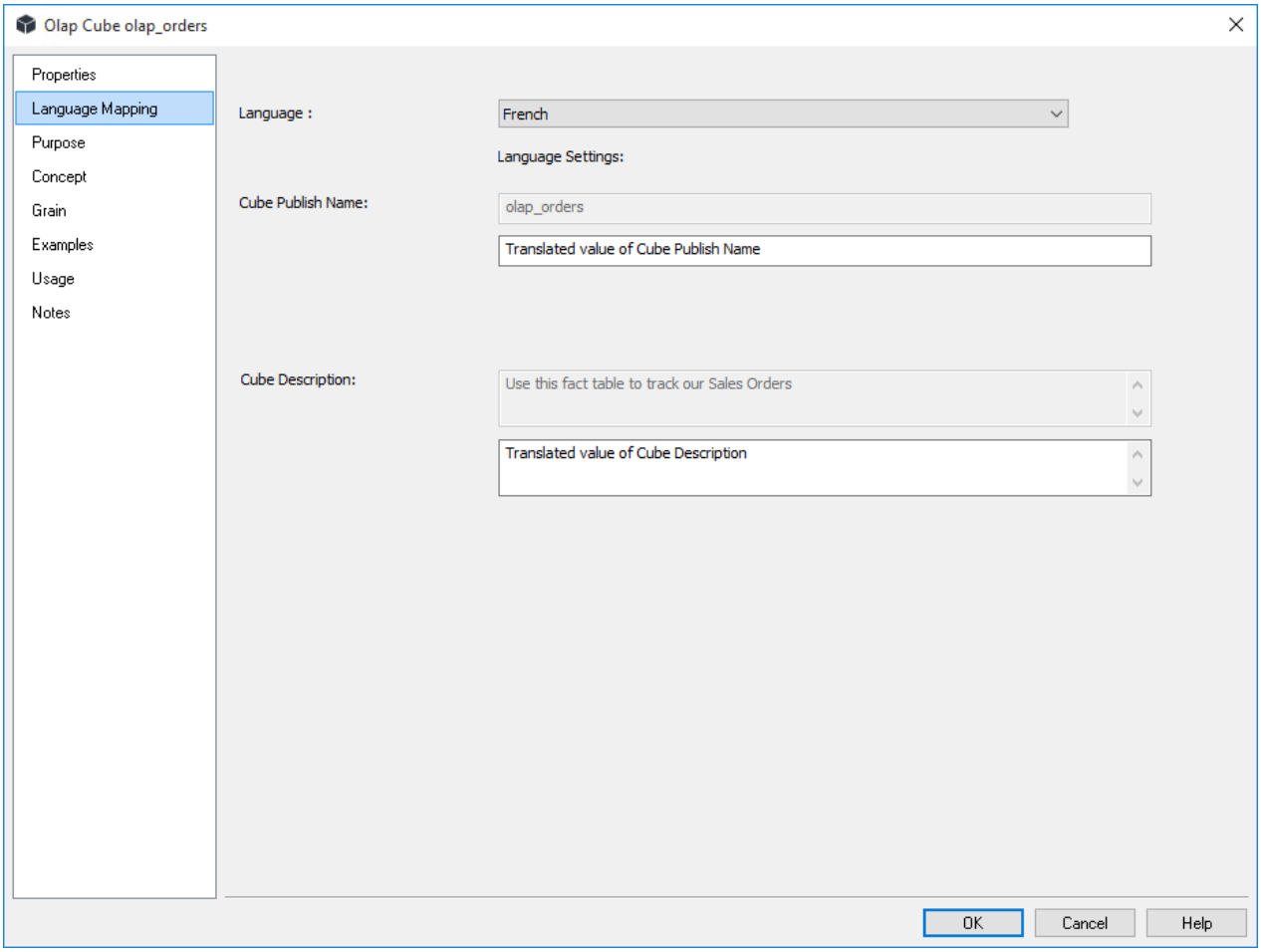
Fields	Description
Storage Mode	This field enables two options; MOLAP (Multidimensional OLAP) or ROLAP (Relational OLAP). At the cube properties level, setting this field determines the defaults for the Storage Mode field ,on its related Measure Groups and partitions.
Default Measure	Specifies the measure used to resolve MDX expressions if a measure is not explicitly referenced in the expression. If no default measure is specified an arbitrary measure is used as the default measure.
Estimated Rows	Specifies the estimated number of rows in the Fact tables. Enter the size of the fact or source table if known, otherwise leave as zero.
Visible	Indicates whether the cube is visible to client applications. • ROLAP stands for Relational Online Analytical Processing. • ROLAP is an alternative to the MOLAP (Multidimensional OLAP) technology. While both ROLAP and MOLAP analytic tools are designed to allow analysis of data using a multidimensional data model, ROLAP differs significantly in that it does not require the pre-computation and storage of information. Instead, ROLAP tools access the data in a relational database and generate SQL queries to calculate information at the appropriate level when an end user requests it. With ROLAP, it is possible to create additional database tables (summary tables or aggregations) which summarize the data at any desired combination of dimensions. • While ROLAP uses a relational database source, generally the database must be carefully designed for ROLAP use. A database which was designed for OLTP will not function well as a ROLAP database. Therefore, ROLAP still involves creating an additional copy of the data. However, since it is a database, a variety of technologies can be used to populate the database.

© Copyright, WhereScape Software Ltd All rights reserved.

Language Mapping

The OLAP Cube Properties screen has a tab called **Language Mapping**.

Select the language from the drop-down list and then enter the translations for the **Cube Publish Name** and the **CubeDescription** in the chosen language.



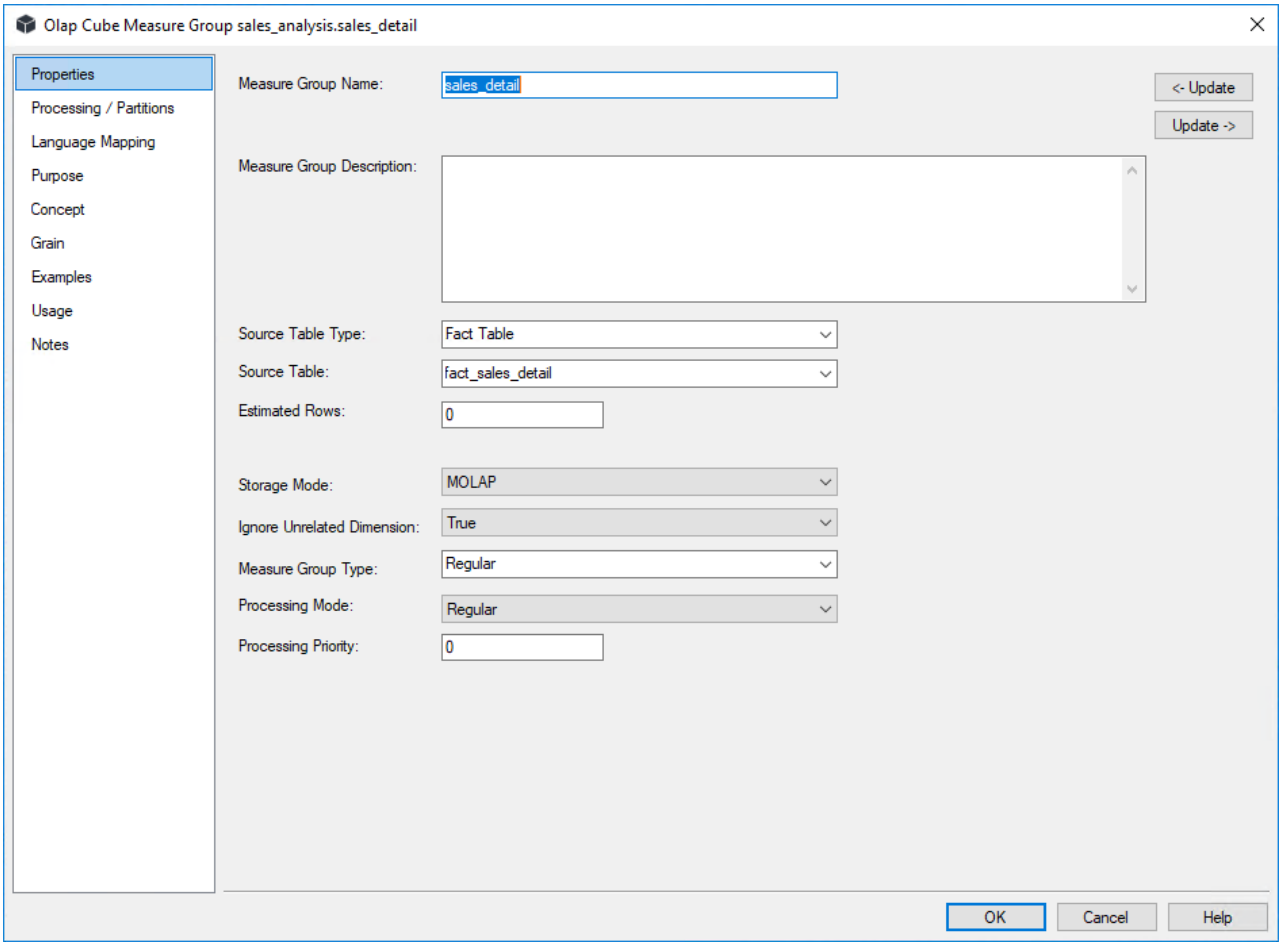
Refer to **Language Options** for details on how to add languages for translation.

© Copyright, WhereScape Software Ltd All rights reserved.

OLAP Cube Measure Groups

A cube can contain multiple measure groups. In WhereScape RED each measure group can belong to a single cube, and each measure group relates to a single star schema. The Measure Groups Cube processing is defined in the tab entitled **Measure Group Properties**.

The Measure Group Properties are shown by right-clicking the Measure Group and choosing **Properties**. The Measure Group Properties associated with cubes are described below.



Field	Description
Measure Group Name	Specifies the name of the Measure Group in Analysis Services
Measure Group Description	Specifies the Metadata description of the Measure Group. This description is stored in Analysis Services and is also used in the WhereScape RED auto-generated documentation.
Update Buttons	The Update Buttons: **Update <-** and **Update ->** are used to move from the current Measure Group to the previous and next Measure Group respectively for the current OLAP cube. The alternative is to exit the Measure Group and choose the next Measure Group Properties screen.
Source Table Type	Specifies the type of table from which the Measure Group has been built in WhereScape RED.
Source Table	Specifies the table from which the Measure Group has been built and populated in WhereScape RED.
Estimated Rows	Specifies the estimated number of rows in the source table. If it is unknown, then leave this value as 0.
Storage Mode	This field provides two options; MOLAP (Multidimensional OLAP) or ROLAP (Relational OLAP). The default value is inherited from the value set at the Cube level. Again, setting this field at the Measure Group level determines the default for the Storage Mode field

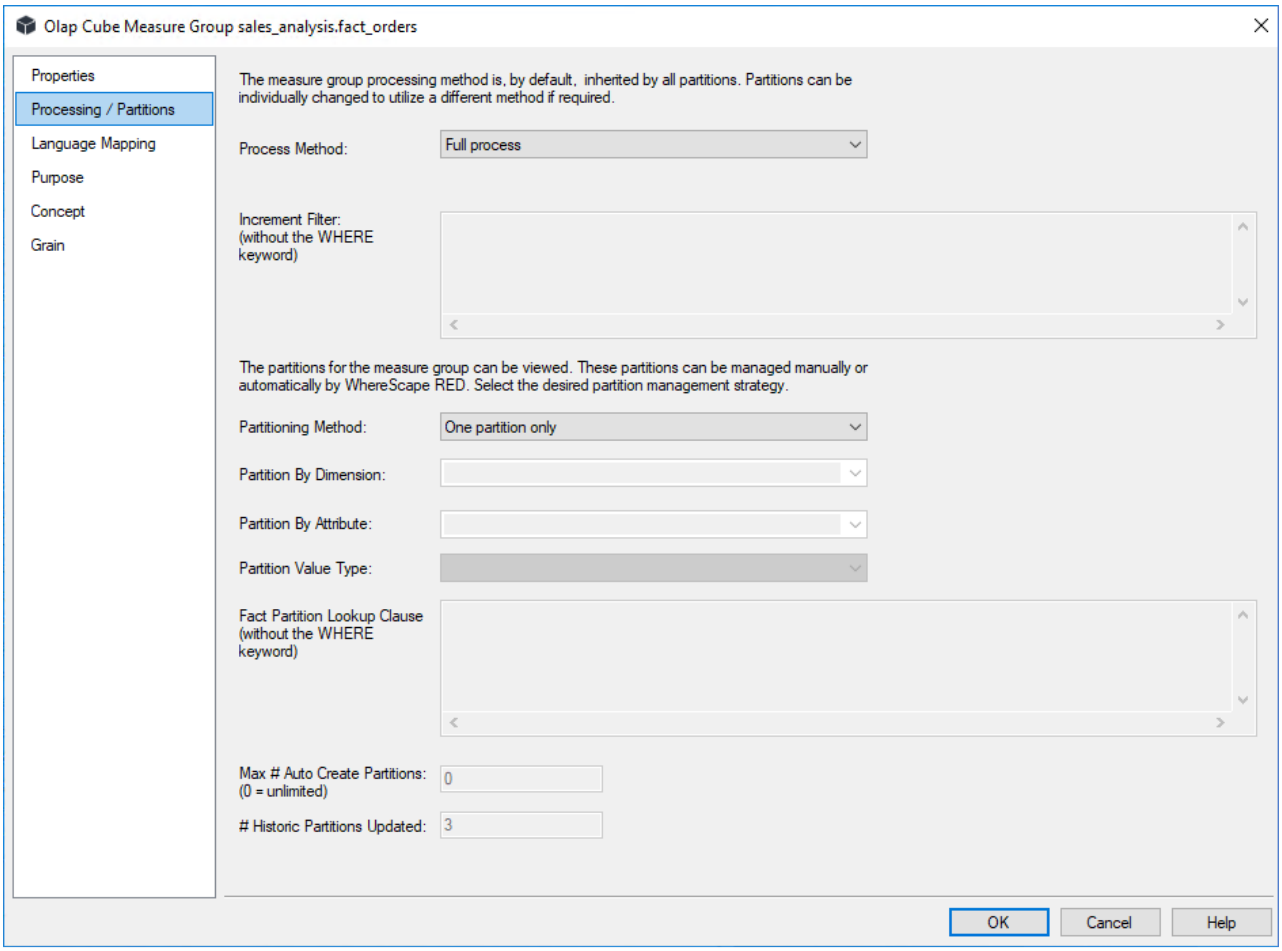
© Copyright, WhereScape Software Ltd All rights reserved.

Field	Description
	on the related Partitions.
Ignore Unrelated Dimensions	Indicates whether dimensions that are unrelated to the Measure Group are forced to their top level when their members are included in a query.
Measure Group Type	Specifies the type of information the Measure Group contains. In some cases, this enables specific treatment by the server and client applications.
Processing Mode	Indicates whether processed data is available after all the aggregations have been computed (Regular) or immediately after the data has been loaded (Lazy Aggregations).
Processing Priority	Specifies the priority for processing the measure group.

OLAP Cube Measure Group Processing/Partitions

Partitions define separately manageable data slices of the measure group data. Partitions can be created, processed and deleted independently within a Measure Group. Each Measure Group needs at least one partition to be defined to enable the Measure Group to be processed.

Partitions are managed through the **Processing/Partitions** tab of the Measure Group's Properties within WhereScape RED.



Field	Description
Process Method	The processing or updating method of a Measure Group is an area that requires careful consideration. The default option is **Full process** which results in the Measure Group being rebuilt. This is in many ways the safest option, but processing time may mean other options must be chosen. The following describes the processing methods that are available in Analysis Services for Measure Groups: • **Default Process** - Detects the process state of the measure group, and performs processing necessary to deliver a fully processed state.

© Copyright, WhereScape Software Ltd All rights reserved.

Field	Description
	• **Full Process** - Processes an Analysis Services Measure Group regardless of state. When Process Full is executed against a Measure Group that has already been processed, Analysis Services drops all data, and then processes it. • **Incremental Process** - Adds newly available fact data and process only to the relevant partitions. To use this option you must set the 'incremental filter' field with a statement that results in only new data being selected. Failure to do so , results in duplicated data in the cube. In many data warehouses a fact table undergoes changes to the data, as well as the addition of new data, and so the incremental update option is not possible. A validation regime must be put in place to compare cube data to the fact data if incremental is used. This validation regime must be used to notify the administrator in the event that duplicate data is inserted into the cube. • **Update Data** - Processes data only without building aggregations or indexes. If there is data in the partitions, it is be dropped before re-populating the partition with source data. This processing option is supported for dimensions, cubes, measure groups, and partitions. • **Build Structure** - If the cube is unprocessed, Analysis Services processes, if it is necessary, all the cube's dimensions. After that, Analysis Services creates only cube definitions. The build structure option just builds the cube structure without populating it. This can be useful if you have a very large cube and want to validate the design.
Increment Filter	If an incremental processing option is chosen, then a filter statement must be selected to only return those rows that are to be added to the Measure Group. As mentioned above, caution must be taken when using incremental updates. For example, if the Measure Group is accidentally processed twice and the filter is based on date, then duplicate data is inserted into the Measure Group without any warning.
Partitioning	Partitioning is useful for handling large datasets where it is impractical to reprocess the entire Measure Group. In such a case, the full process option would probably be chosen but only selected partitions would be processed. See the section on partitioning for more information. The default process performs a full process on the first pass followed by incremental updates on subsequent processing runs. Caution must be taken when choosing default for the cube.
Partitioning Method	Three options are provided for handling Measure Group partitions. They are: 1. **One partition only** - When this option is selected the partition information for the Measure Group is ignored and one partition is created and processed for the Measure Group. This would be the normal situation unless performance issues require an alternate strategy. 2. **Manually managed multiple partitions** - With this option the partition information stored for the Measure Group is used in the creation and processing of the Measure Group. 3. **Automatic partition handling** - This option is available if the Measure Group is to be partitioned by one numeric value. The partitioning should preferably be on something like day, month or year. (i.e. YYYY, YYYYMM or YYYYDDD). If this option is chosen together with one of the date formats described above, then WhereScape RED automatically creates partitions as required, and process only those partitions that are marked for processing.
Partition by Dimension	This field is only available if automatic partition handling is chosen. Select the dimension in the Measure Group that we will partition by. This would normally be a date dimension.
Partition by Attribute	This field is only available if automatic partition handling is chosen. Select the attribute that we are to partition by. This would normally be a year or maybe a month level. (e.g. cal_year, fin_year from the WhereScape date dimension).
Partition by Value Type	This field is only available if automatic partition handling is chosen. Select the type of level we are dealing with. Choose YYYY for a year partition and YYYYMM for a month partition. This format must correspond with the column in the date dimension.

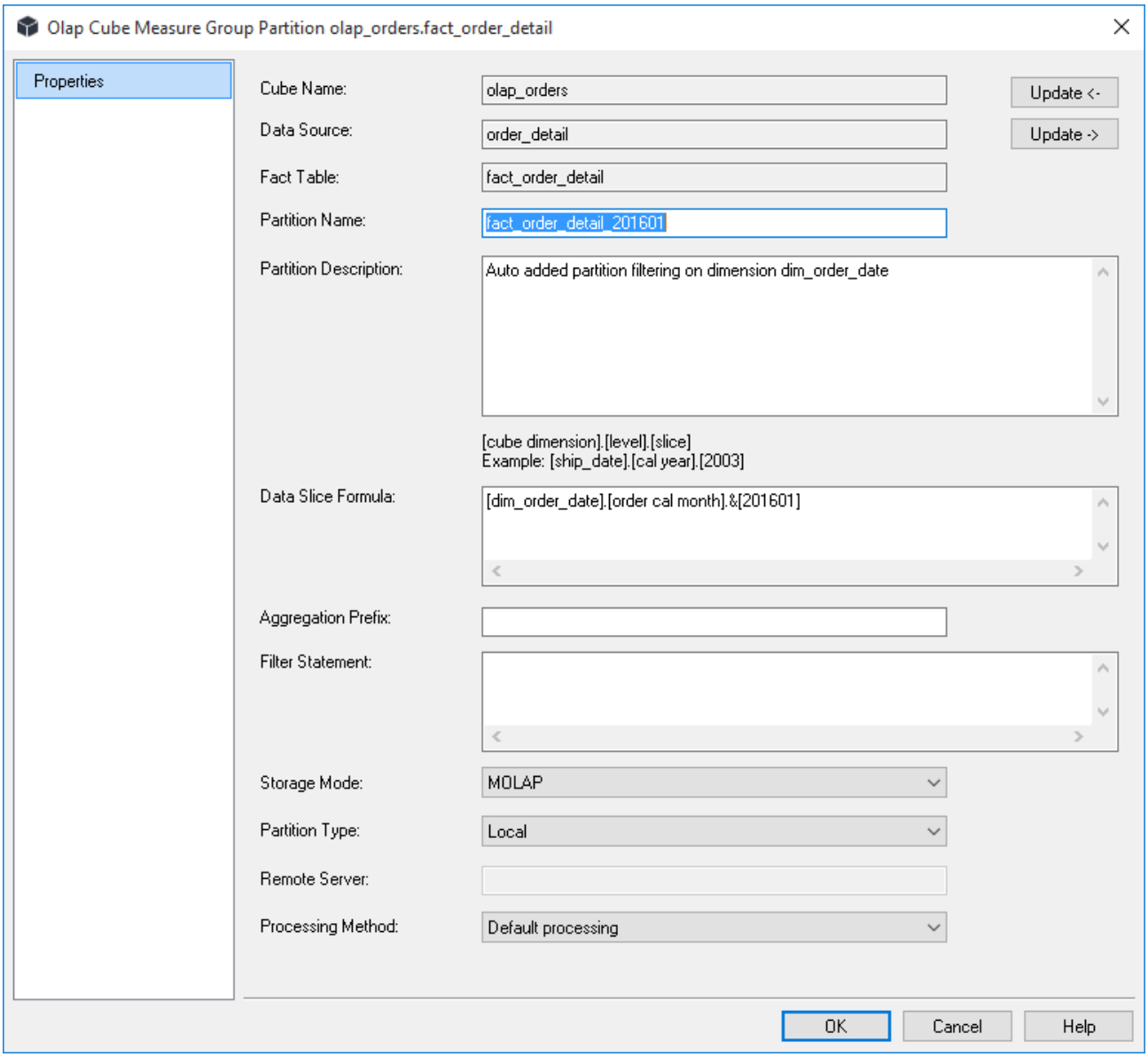
© Copyright, WhereScape Software Ltd All rights reserved.

Field	Description
	WhereScape RED only supports partitioning by Year, Quarter, month or day.
Fact Partition Lookup Clause	This field is only available if automatic partition handling is chosen. In order to know when to create a new partition WhereScape RED executes a query against the fact table and the date dimension to acquire each unique period. When dealing with a large fact table, such a query may take a long time to complete. This field can be used to include the components of the 'Where' clause to restrict the amount of data examined. For example, we may enter 'dss_update_time < GETDATE()-14' to only look at fact table records that have been inserted or updated in the last 14 days. This still enables you to catch any new partitions and add them. The first time a cube is converted to auto partitioning handling, a full pass of the Fact table should occur to allow inclusion of every partition. This field must therefore only be populated once the cube has been initially built with all partitions intact.
Max Number of Auto Created Partitions	This field is only available if automatic partition handling is chosen. You can specify an upper limit for automatically created partitions. The default is zero, or no limit. This limit may be useful if your source system can get erroneous data. If set, then the processing of the Measure Group will fail if a new partition will exceed the counter.
Number of Historic Partitions Updated	This field is only available if automatic partition handling is chosen. This field enables you to restrict the partition updating to the latest nnn partitions. If for example, you were partitioning by year and set this value to 2, you would process the current and previous years only. WhereScape RED turns OFF partition processing, after it does a partition update, so the first pass still updates all partitions.
To Display Measure Groups	To display a list of measure groups defined against a cube, right-click on a cube and select **Display Measure Groups**.
To Add a Measure Group	To add a measure group, display the measure groups in the middle pane and either: • Drag over a new fact table into the target pane - this automatically creates a new measure group in the cube. Any additional dimensions required to support analysis of the Measure Group is added to the cube. • Right-click the cube in the Objects list pane and then select **Add Measure Group** and fill in the Measure Group properties.
To Delete a Measure Group	To delete a measure group, display the measure groups in the middle pane and then right-click to select delete measure group.
Displaying Measures	Measures can be displayed or added while viewing measure groups in the middle pane. Right-click a measure group and select the appropriate option.
Displaying Partitions	Partitions can be displayed or added while viewing measure groups in the middle pane. Right-click a measure group and select the appropriate option.

© Copyright, WhereScape Software Ltd All rights reserved.

OLAP Cube Measure Group Partitions

The Measure Group Partition's properties are shown by right-clicking the Measure Group Partition displayed in the middle pane and then selecting **Properties**. The partition's properties associated with a measure group are described below.



Fields	Description
Cube Name	The name of the cube that the partition belongs.
Data Source	The data source for the partitions. This will be inherited from the cube and cannot be changed. You cannot have partitions with different data sources or different fact tables in WhereScape RED. If you need to support either scenario, then the partition must be created directly within Analysis Services. In such a case, it can still be managed in terms of processing through WhereScape RED.
Fact Table	The fact table that the data is derived from. This is inherited from the cube and cannot be changed. See the notes above under data source.
Update Buttons	The Update Buttons: **Update <-** and **Update ->** are used to move from the current Measure Group Partition to the previous and next Measure Group Partition respectively for the current Measure Group. The alternative is to exit the Measure Group Partition and choose the next Measure Group Partition Properties screen.

© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
Partition Name	Where only one partition exists, it is normally given the same name as the cube. If manually creating then a unique name must be assigned for each partition. If auto partitioning is chosen, then WhereScape RED uses the cube name plus the level value to make the partition name.
Partition Description	A description of the partition for documentation purposes.
Data Slice Formula	This field defines the range of data stored in the partition. It is a very simplified version of what can be done in Analysis Services. If a more complex partitioning algorithm is required then the partition must be created in Analysis Services. The format for the formula is as follows: The brackets must surround each name and a full stop must separate the three parts of the formula. For example, a cube that is partitioned by year on its date dimension would have the following formula for the 2003 year: [dim_date].[cal_year].[2003]
Aggregation Prefix	By default, any cube aggregation is prefixed with the partition name. An alternate name can be entered here. Refer to Analysis Services for details.
Filter Statement	Not implemented.
Storage Mode	This field has two options; MOLAP (Multidimensional OLAP) or ROLAP.(Relational OLAP). This determines how the OLAP cube is processed. The default value is inherited from the value set at the Measure Group level.
Partition Type	The partition can be either Local or Remote. Local means that the partition resides on the same Analysis Services server as the cube. If Remote is chosen then a server must be specified where the partition will be located.
Remote Server	If a Remote partition then the name of the remote Analysis Services server must be entered here.
Processing Method	A partition is either enabled for processing or disabled. This field can be set to Always process or Never process. If left as default and in automatic mode, then WhereScape RED deactivates the processing, once the partition has been aged out.

© Copyright, WhereScape Software Ltd All rights reserved.

OLAP Cube Measures

Measures represent the numeric attributes of a fact table that are aggregated, using an OLAP aggregate function defined against each Measure. Each Measure is defined against a Measure Group, which is defined against a cube. The properties of a measure are shown by right-clicking a Measure and the selecting **Properties**.

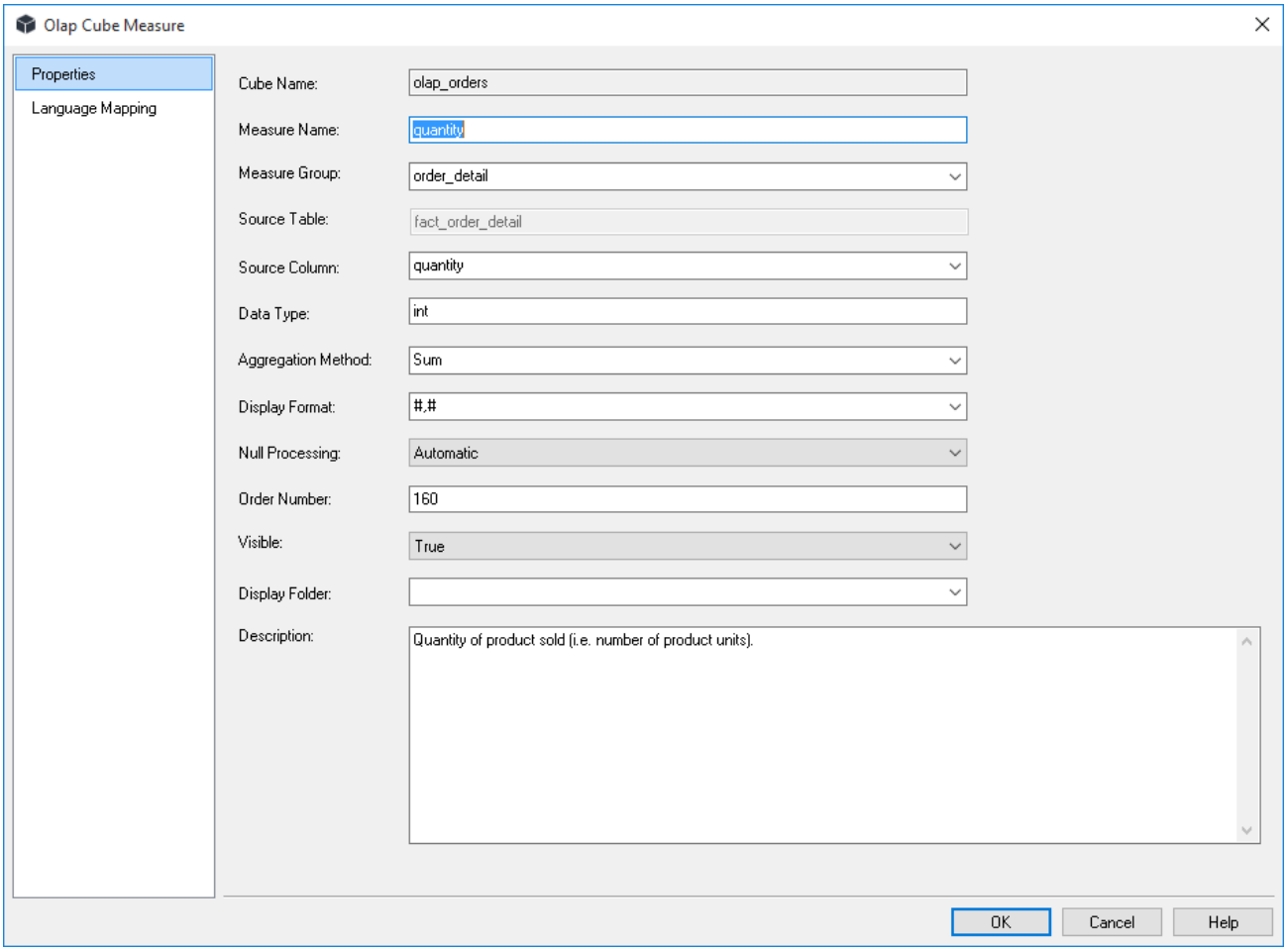


Table	Fields
Cube Name	A Read Only field that indicates against which OLAP Cube the measure is defined.
Measure Name	Specifies the Analysis Services name defined for the measure.
Measure Group	Specifies against which Measure Group the Measure is defined. This is related to the fact table of which the Measure is an attribute.
Update Buttons	The Update Buttons: **Update <-** and **Update ->** are used to move from the current Measure to the previous and next Measure respectively for the current OLAP Cube Measure Group. The alternative is to exit the Measure and choose the next Measure Properties screen.
Source Table	A read only field that indicates the fact table that is related to the Measure Group.
Source Column	Specifies from which numeric attribute of the underlying fact table the Measure is built. This is a drop-down list populated from the RED metadata definition of the fact table associated with the Measure Group above.
Data Type	Specifies the data type used by Analysis Services. The data type specified for this property is inherited from the fact table attribute defined in WhereScape RED metadata but can be different (typically a larger data type is used in the cube to cope with the larger numbers generated by aggregating the source data).

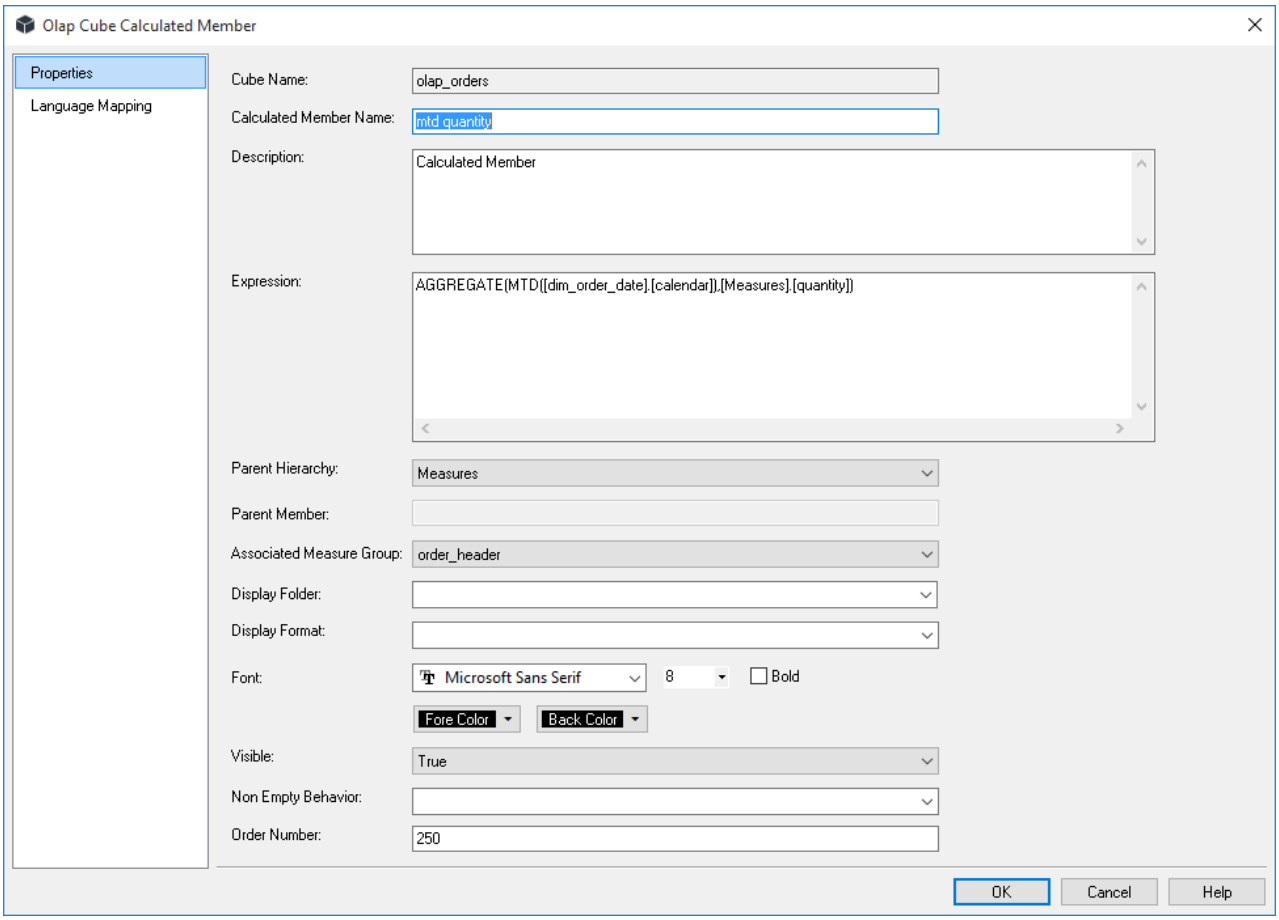
© Copyright, WhereScape Software Ltd All rights reserved.

Table	Fields
Aggregation Method	Specifies the OLAP function used to aggregate measure values. The default options are: • Sum • Count • Distinct Count - only one distinct count is allowed per Measure Group and can have query performance implications. • Min - Minimum • Max - Maximum
Display Format	Specifies the format used by clients when displaying the measure value.
Null Processing	Specifies the processing null values. Setting this property to Automatic means that Analysis Services uses default behavior.
Order Number	The order in which the measures appear in the cube is dictated by their order number.
Visible	Measures are normally visible, but some measures used in calculations may be hidden. In such a case, clear this check box.
Display Folder	Cube Measures can be organized into user-defined folders to view and manage these attributes within the Analysis Services user interface more easily. Enter the display folder name. **Note:** One object can be in multiple display folders, for example: Display Folder: financials;sales
Description	A description of the measure which is stored in the cube. This description by default is acquired from the source column.
To View measures	The measures can be viewed by clicking an OLAP Cube in the left pane, which displays the measures in the middle pane.
To Add a New Measure	To add a new measure, view the measures in the middle pane, right-click in the middle pane and select add measure. This can also be done when viewing measure groups in the middle pane. Alternatively to create measure which is very similar to an existing measure, view the measures in the middle pane and right-click, select **Copy Measure**. The same dialog box appears as for **Add Measure** with most of the fields filled in. Notice that the measure name has the suffix "- Copy". Change the name in the Measure Name field and make any other alterations and then click **OK**.
To Delete a measure	Display the measures in the middle pane, select the measure to delete, right-click and select **Delete**.

© Copyright, WhereScape Software Ltd All rights reserved.

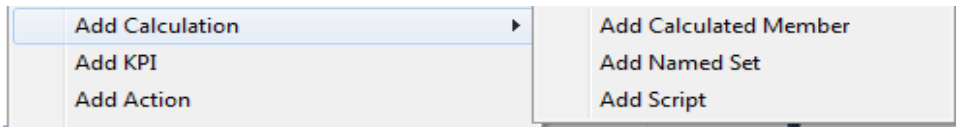
OLAP Cube Calculations

Calculations provide the ability to define the derivation of a value at query time within a cube. The calculation is most typically a numeric derivation based on measures, but can be defined against any dimension. The calculation is defined in MDX (Multi-Dimensional eXpressions). The definition of a Calculation is shown by right-clicking a **Calculation** and choosing **Properties**. The following Cube Calculated Member Definition window is displayed:



Fields	Description
Cube Name	A Read Only field that indicates against which OLAP Cube the Calculation is defined.
Calculated Member Name	Specifies the Analysis Services name defined for the Calculation.
Update Buttons	The Update Buttons: **Update <-** and **Update ->** are used to move from the current Calculation to the previous and next Calculation respectively for the current OLAP Cube. The alternative is to exit the Calculation and choose the next Calculation Properties screen.
Description	A business description of the Calculation that is used to populate WhereScape RED documentation.
Expression	Specifies the MDX expression that defines the calculation.
Parent Hierarchy	Specifies where the calculation is displayed for use. By default and in most cases, this will be 'Measures'. This means that the calculated member is displayed to the end user of the cube as a measure, otherwise known as a calculated measure. Alternatively, you can include the calculated member in a dimension instead of in the measures. Hierarchies are descriptive categories of a dimension by which the measures in a cube can be separated for analysis. A calculated member provides a new label in the parent dimension you select.
Parent Member	This is not available if you select **Measures** as your parent hierarchy, or if you select a one-level hierarchy. Hierarchies are divided into levels that contain members. Each member produces a heading in the cube. While browsing a cube, users can drill down to

© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
	subordinate headings. The heading for the calculated member is added directly below the selected Parent Member.
Associated Measure Group	Specifies against which Measure Group the Measure is defined. This is related to the fact table of which the Measure is an attribute. **Note:** SSAS 2008+ uses the property **Associated Measure Group** but previous versions of SSAS do not and can result in errors when creating the cube. It is possible to set this attribute in RED to **(Undefined)** for previous versions of RED, but this is not necessary for the current version of RED as this attribute will only be used when appropriate.
Display Folder	Cube Calculations can be organized into user-defined folders to view and manage these attributes within the Analysis Services user interface more easily. Enter the display folder name. **Note:** One object can be in multiple display folders, for example:
Display Format	Specifies the format used by clients when displaying the measure value.
Visible	Specifies whether the calculation is visible to client tools.
Non Empty Behavior	Determines the non-empty behavior associated with the calculation.
Order Number	The create order of the member in the dimension hierarchy.
To View Calculations	To view the list of calculations (sometimes called a calculated measure), right-click the OLAP Cube in the left pane and then select **Display Calculations**.
To Add a Calculation	1. To add a calculation, right-click the OLAP Cube in the left pane, select **Add Calculation** and then choose between: • Add Calculated Member • Add Named Set • Add Script  This can also be achieved by displaying calculations in the middle pane, right-clicking and selecting **Add Calculated Member**, **Add Named Set** or **Add Script**. Fill out the dialog box with the relevant details. 2. To create a calculation that is similar to an existing calculation, display the calculations in the middle pane and select **Copy Calculation**. The same dialog box appears as for **Add Calculation** with most of the fields filled in. Notice that the calculated member name has the suffix "- Copy". Change the name in the Measure Name field and make any other alterations and then click **OK**.
To Delete a calculation	Display the calculations in the middle pane, select the calculation to delete, right-click and select **Delete**.

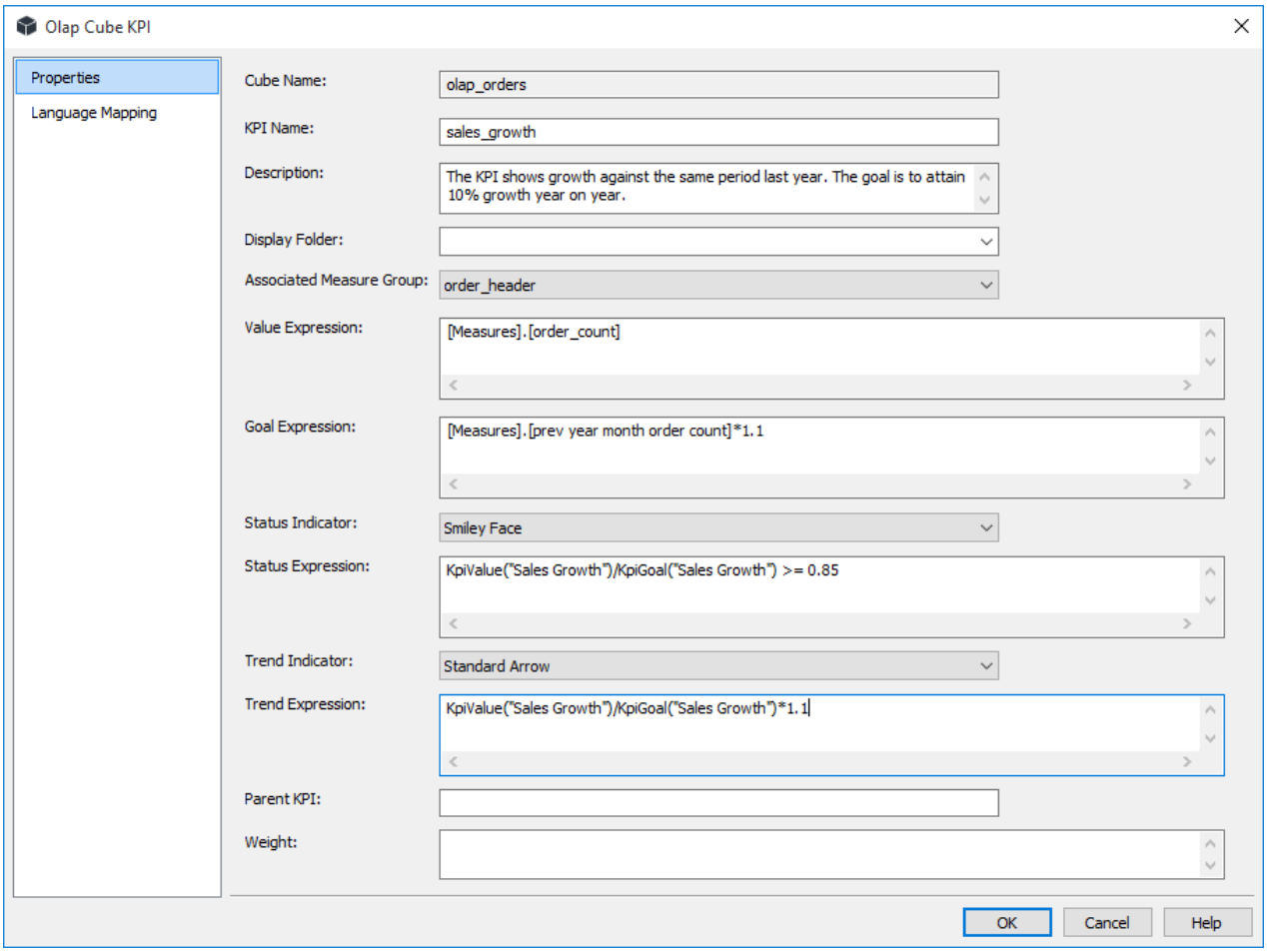
© Copyright, WhereScape Software Ltd All rights reserved.

OLAP Cube Key Performance Indicators

In Analysis Services, a KPI is a collection of calculations that are associated with a measure group in a cube that are used to evaluate business success. Typically, these calculations are a combination of Multidimensional Expressions (MDX) expressions or calculated members.

KPIs also have additional metadata that provides information about how client applications display the results of the KPI's calculations. The definition of a Calculation is shown by right-clicking a Calculation and choosing **Properties**.

The following Cube KPI Definition window is shown below:



Fields	Description
Cube Name	A Read Only field that indicates against which OLAP Cube the KPI is defined.
KPI Name	Specifies the name of the KPI defined in Analysis Services.
Description	A business description of the KPI that is used to populate WhereScape RED documentation.
Update Buttons	The Update Buttons: **Update <-** and **Update ->** are used to move from the current KPI to the previous and next KPI respectively for the current OLAP Cube. The alternative is to exit the KPI and choose the next KPI Properties screen.
Display Folder	KPIs can be organized into user-defined folders to view and manage these attributes within the Analysis Services user interface more easily. Enter the display folder name. **Note:** One object can be in multiple display folders, for example: Display Folder: financials;sales

© Copyright, WhereScape Software Ltd All rights reserved.

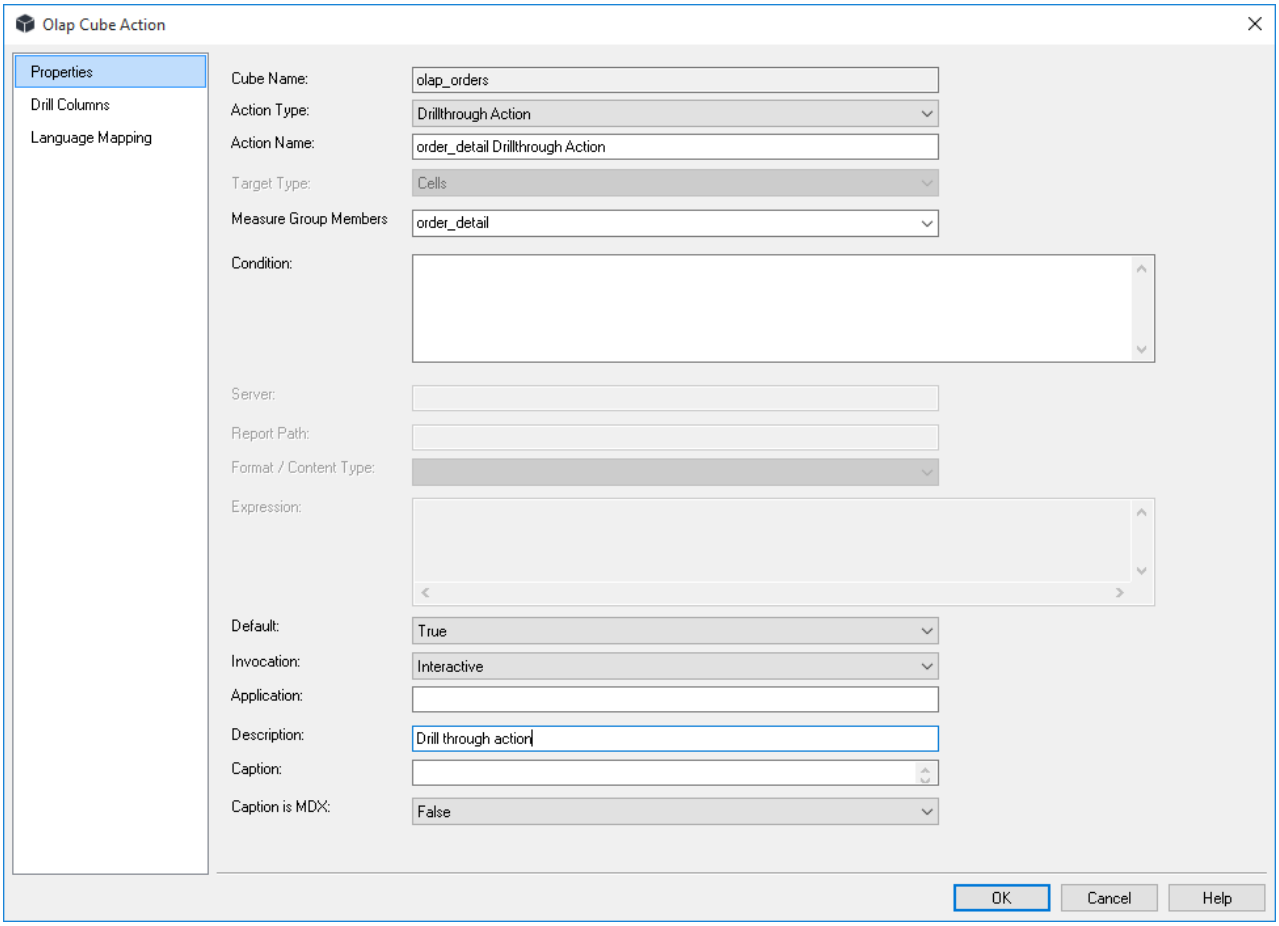
Fields	Description
Associated Measure Group	Specifies the Measure Group against which the KPI is defined.
Value Expression	An MDX numeric expression that returns the actual value of the KPI.
Goal Expression	An MDX numeric expression or a calculation that returns the target value of the KPI.
Status Indicator	A visual element that provides a quick indication of the status of a KPI. The display of the element is determined by the value of the MDX expression that evaluates status.
Status Expression	An MDX expression that represents the state of the KPI at a specified point in time. • The status MDX expression should return a normalized value between -1 and 1. Values equal to or less than -1 is interpreted as "bad" or "low." A value of zero (0) is interpreted as "acceptable" or "medium." Values equal to or greater than 1 is interpreted as "good" or "high." • An unlimited number of intermediate values can optionally be returned and can be used to display any number of additional states, if supported by the client application.
Trend Indicator	A visual element that provides a quick indication of the trend for a KPI. The display of the element is determined by the value of the MDX expression that evaluates trend.
Trend Expression	An MDX expression that evaluates the value of the KPI over time. The trend can be any time-based criterion that is useful in a specific business context. The trend MDX expression enables a business user to determine whether the KPI is improving over time or degrading over time.
Parent KPI	A reference to an existing KPI that uses the value of the child KPI as part of computation of the parent KPI. Sometimes, a single KPI is a computation that consists of the values for other KPIs. This property facilitates the correct display of the child KPIs underneath the parent KPI in client applications.
Weight	An MDX numeric expression that assigns a relative importance to a KPI. If the KPI is assigned to a parent KPI, the weight is used to proportionally adjust the results of the child KPI value when calculating the value of the parent KPI.
To View KPIs	To view the list of KPIs, right-click the OLAP Cube in the left pane and select **Display KPIs**.
To Add a KPI	1. To add a calculation, right-click the OLAP Cube in the left pane and select **Add KPI**. This can also be achieved by displaying KPIs in the middle pane, right-clicking and selecting **Add KPI**. Fill out the dialog box with the relevant details. 2. To create a KPI that is similar to an existing KPI, display the KPIs in the middle pane and select **Copy KPI**. The same dialog box appears as for **Add KPI** with most of the fields filled in. Notice that the KPI name has the suffix "- Copy". Change the name in the **KPI Name** field and make any other alterations and then click **OK**.
To Delete a KPI	Display the KPIs in the middle pane, right-click the KPI and then select **Delete**.

© Copyright, WhereScape Software Ltd All rights reserved.

OLAP Cube Actions

An action provides information to a client application to allow an action to occur based on the property of a clicked dimensional member. Actions can be of different types and they have to be created accordingly.

To view the definition of an Action, right-click the **Action** and select **Properties**. The **OLAP Cube Action** window is displayed:



Fields	Description
Cube Name	A Read Only field that indicates against which OLAP Cube the Calculation is defined.
Action Type	Actions can be of the following types: • **Drill through actions** which return the set of rows that represents the underlying data of the selected cells of the cube where the action occurs. When this option is chosen an additional tab is enabled that allows drill through columns to be chosen. • **Reporting actions** which return a report from Reporting Services that is associated with the selected section of the cube where the action occurs. • **Standard actions** (Action), which return the action element (URL, HTML, DataSet, RowSet, and other elements) that is associated with the selected section of the cube where the action occurs. The action type chosen determines which action specific fields are enabled within the dialog.
Action Name	Defines the name of the Action in the Cube.
Update Buttons	The Update Buttons: **Update <-** and **Update ->** are used to move from the current Action to the previous and next Action respectively for the current OLAP Cube. The alternative is to exit the Action and choose the next Action Properties screen.
Target Type	Select the object to which the action is attached. Generally, in client applications, the action is displayed when end users select the target object; however, the client

© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
	application determines which end-user operation displays actions. For Target type, select from the following objects: • Attribute members • Cells • Cube • Dimension members • Hierarchy • Hierarchy members • Level • Level members
Target Object	The cube object of the designated target type against which the action is defined.
Condition	Specify an optional Multidimensional Expressions (MDX) expression that resolves to a Boolean value. If the value is True, the action is performed on the specified target. If the value is False, the action is not performed.
Report Action: Server	The name of the computer running report server.
Report Action: Report Path	The path exposed by report server.
Format/Content Type	Select the type of action. The following table summarizes the available types: • Data Set - Retrieves a dataset • Proprietary - Performs an operation by using an interface other than those listed in this table. • Row Set - Retrieves a rowset. • Statement - Runs an OLE DB command. • URL - Displays a page in an Internet Browser.
Expression	Specifies the parameters that are passed when the action is run. The syntax must evaluate to a string, and you must include an expression written in MDX. MDX expressions are evaluated before the parameters are passed.
Default	An additional true/false drop-down list is enabled for Drill through actions. This gets or sets the current Drillthrough Action as the default action, when multiple drill through actions are defined. This is important for users of MS Excel 2007 to browse the Olap Cube because MS Excel only invokes the Drill through Action marked as the default.
Invocation	Specifies how the action is run. Interactive, the default, specifies that the action is run when a user accesses an object. The possible settings are: • Batch • Interactive • On Open
Application	Describes the application of the action.
Description	Describes the action.
Caption	Provides a caption that is displayed for the action.
Caption is MDX	If the caption is MDX, specify True, if not specify False.
To View Actions	To view the list of Actions, right-click on an OLAP Cube in the left pane and select **Display Actions**.
To Add an Action	1. To add an Action, right-click on an OLAP Cube in the left pane and select **Add Action**. This can also be achieved by displaying Actions in the middle pane, right-clicking and selecting **Add Action**. Fill out the dialog box with the relevant details. 2. To create an Action that is similar to an existing Action, display the Actions in the middle pane and select **Copy Action**. The same dialog box appears as for **Add**

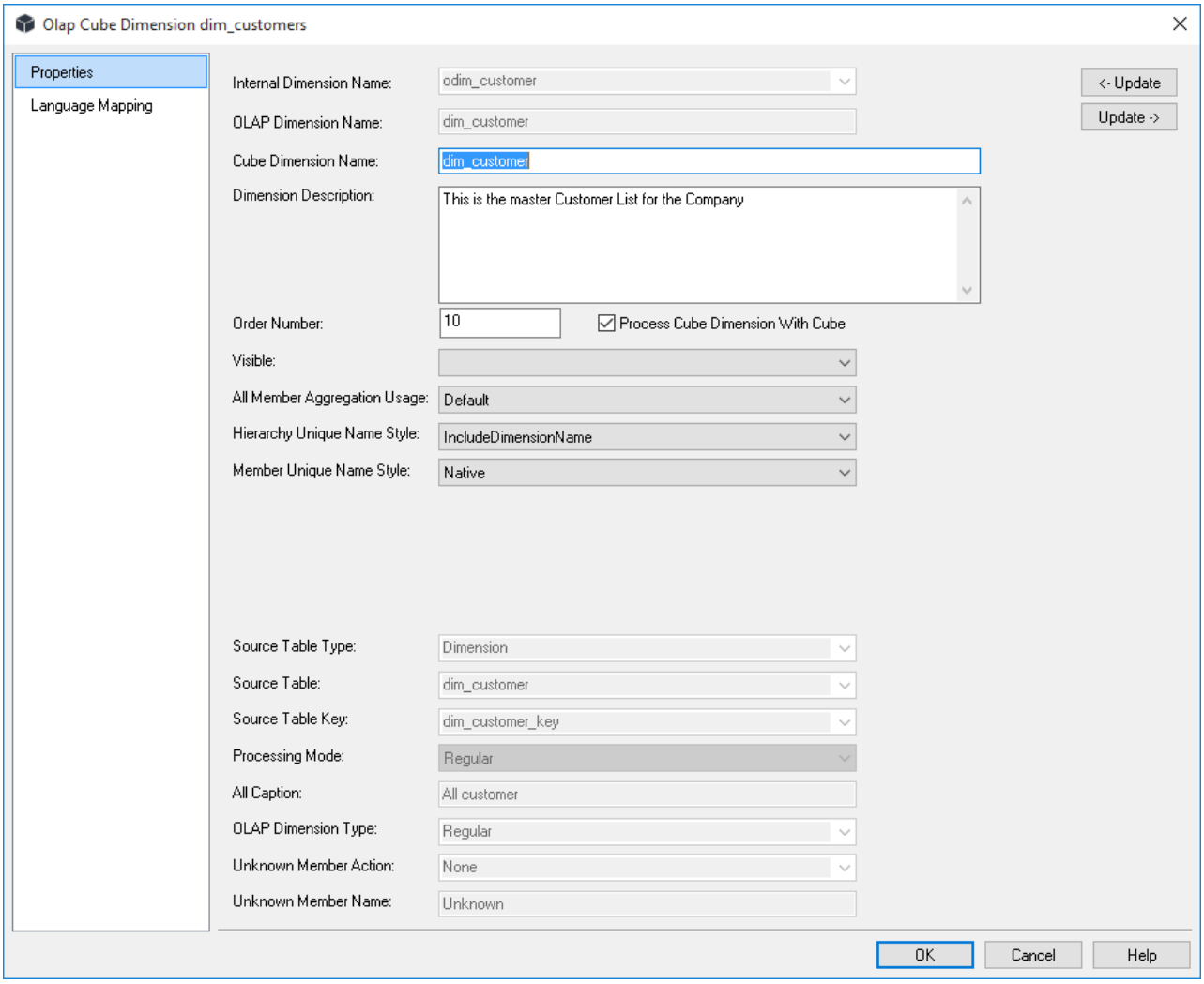
© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
	Action with most of the fields filled in. Notice that the Action name has the suffix "- Copy". Change the name in the Action Name field and make any other alterations and click **OK**.
To Delete an Action	Display the Actions in the middle pane, right-click the Action and then select **Delete**.

OLAP Cube Dimensions

OLAP Dimensions are associated automatically with a cube when a cube is created in WhereScape RED based on the underlying star schema. OLAP Dimensions that are associated with a cube can be displayed, or additional OLAP Dimensions can be manually added from the list of OLAP Dimensions defined in WhereScape RED.

Once an OLAP Dimension is associated with a cube a relationship is created with the relevant Measure Groups within the cube—these relationships are defined automatically with WhereScape RED, and they can also be added. The properties of an OLAP Dimension associated with a cube are shown by right-clicking the cube dimensions listed in the middle pane and selecting **Properties**. The properties are shown below:



Fields	Description
Internal Dimension Name	A read only field displaying the name of the OLAP Dimension in WhereScape RED.
OLAP Dimension Name	Specifies the name of the OLAP Dimension as a Dimension in Analysis Services.

© Copyright, WhereScape Software Ltd All rights reserved.

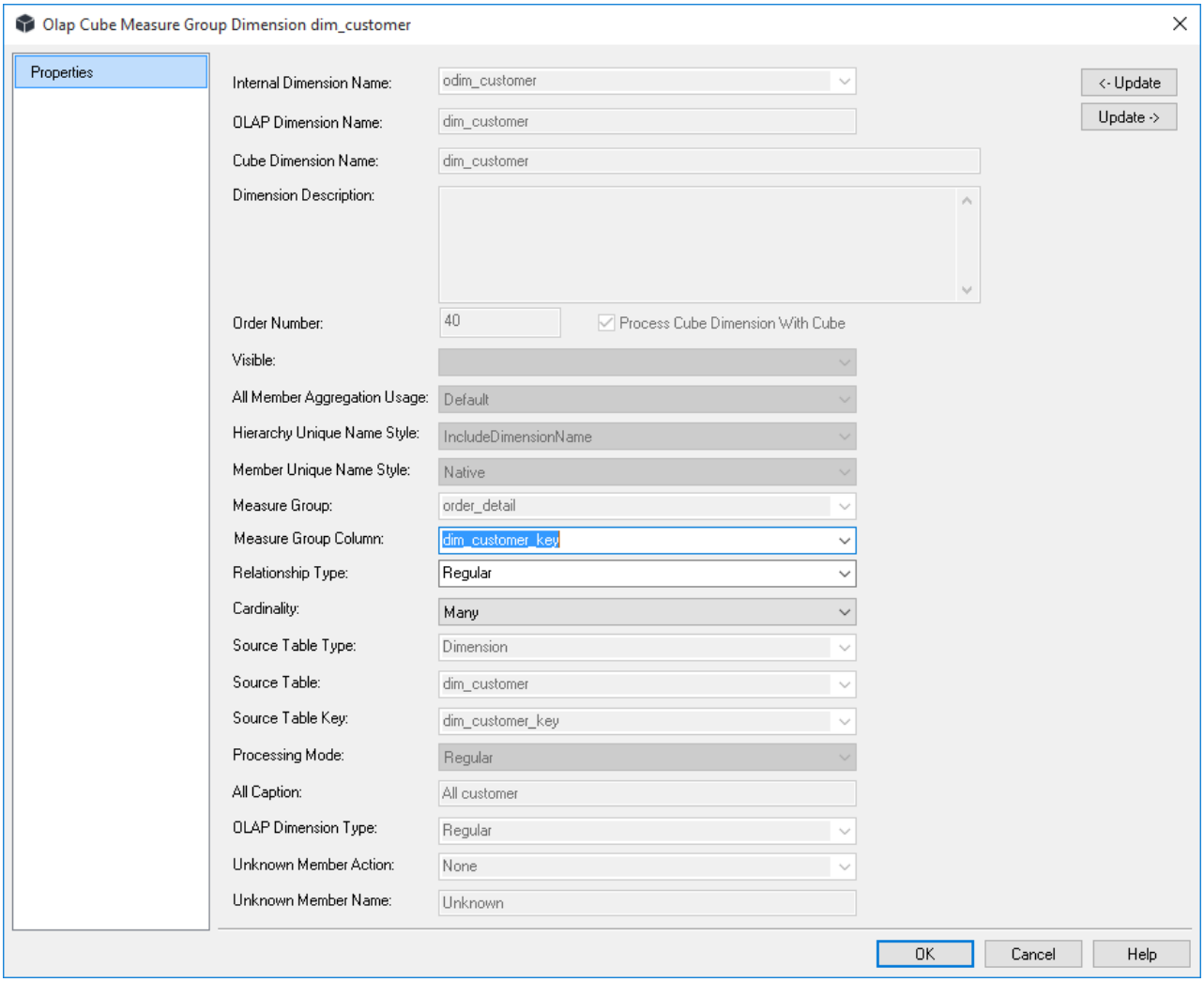
Fields	Description
Cube Dimension Name	Specifies the exposed name of the Dimension when associated with a cube (this can be different from the OLAP Dimension Name).
Update Buttons	The Update Buttons: **Update <-** and **Update ->** are used to move from the current OLAP Dimension to the previous and next OLAP Dimension respectively for the current OLAP Cube. The alternative is to exit the OLAP Dimension and choose the next OLAP Dimension Properties screen.
Dimension Description	A description of the dimension when associated with the cube.
Order Number	The order number.
Process Cube Dimension With Cube	Use this check box to enable the dimension to be processed with the OLAP Cube.
Visible	Determines whether or not the dimension is visible to client applications.
All member Aggregation Usage	Specifies how aggregations will be designed by the BIDS Storage Design Wizard if it is used to design cube aggregations.
Hierarchy Unique Name Style	Indicates whether the dimension name is included in the name of the hierarchies. If set to Default, then the system applies a default behavior and will include the dimension name, in cases where there is more than one usage of the same dimension.
Member Unique Name Style	Indicates how member unique names is formed. Other Read Only fields that are displayed in this window are configurable against the OLAP Dimensions' properties and cannot be changed in this window, including: • Source Table Type • Source Table • Source Table Key • Processing Mode • All caption • OLAP Dimension Type • Unknown Member Action • Unknown Member Name
To View Cube Dimensions	To view the list of Dimensions associated with a cube, right-click the OLAP Cube in the left pane and then select **Display Dimensions**.
To Add a Cube Dimension	To add an existing OLAP Dimension, right-click the OLAP Cube in the left pane and then select **Add Dimension**. This can also be achieved by displaying Dimensions in the middle pane, right-clicking and then selecting **Add Dimension**. Fill out the window with the relevant details.
To Remove a Cube Dimension	Display the Cube Dimensions in the middle pane, select the Dimension to remove, right-click and select **Remove Dimension from Cube**. This action removes the association of the OLAP Dimension from the OLAP Cube.

© Copyright, WhereScape Software Ltd All rights reserved.

OLAP Cube Measure Group Dimensions

Measure group dimensions are the relationships between cube Measure Groups and OLAP Dimensions. In WhereScape RED this equates to the relationships between Fact tables and Dimensions in the underlying star schema.

The **Properties** are shown below:



Fields	Description
Internal Dimension Name	A Read-Only field displaying the name of the OLAP Dimension in WhereScape RED.
OLAP Dimension Name	A Read-Only field displaying the name of the OLAP Dimension as a Dimension in Analysis Services.
Cube Dimension Name	A Read-Only field displaying the name of the Dimension when associated with a cube (this can be different from the OLAP Dimension Name).
Update Buttons	The Update Buttons: **Update <-** and **Update ->** are used to move from the current OLAP Measure Group Dimension to the previous and next OLAP Measure Group Dimension respectively for the current OLAP Cube. The alternative is to exit the OLAP Measure Group Dimension and choose the next OLAP Measure Group Dimension Properties screen.
Dimension Description	A description of the dimension when it is associated with the cube.
Order Number	The order number.

© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
Update Dimension with Cube	A Read-Only check box showing whether the dimension is processed when the cube is processed.
Visible	A Read-Only field displaying the visibility of the dimension on the cube.
All Member Aggregation Usage	A Read-Only field displaying how aggregations will be designed by the BIDS Storage Design Wizard if it is used to design cube aggregations.
Hierarchy Unique Name Style	A Read-Only field which indicates whether the dimension name will be included in the name of the hierarchies.
Member Unique Name Style	A Read-Only field which indicates how member unique names will be formed. A read only field.
Measure Group	A Read-Only field displaying the Measure Group which is being referenced by the Measure Group Dimension relationship.
Measure Group Column	Specifies which fact table key joins to the dimension key.
Relationship Type	Defines the relationship type for the relationship between the Dimension and Measure Group. In WhereScape RED, this option can be Regular, which means that the relationship is based on a dimension key join, or No Relationship between the Measure Group and Dimension.
Cardinality	Indicates whether the measure group has a many to one or one to one relationship with the dimension.
Source Table Type	A Read-Only field that displays the type of table from which the OLAP Dimension was created.
Source Table	A Read-Only field that displays the name of the table from which the OLAP Dimension was created.
Source Table Key	A Read-Only field that displays the key (typically the primary key) that relates the dimension to the fact in the underlying star schema.
Processing Mode	A Read-Only field that indicates whether processed data is available after all aggregations have been computed or immediately after the data has been loaded.
All Caption	A Read-Only field that displays the name of the (All) member. This applies to all hierarchies in the dimension that have an (All) member.
OLAP Dimension Type	A Read-Only field that displays the type of information contained by the dimension.
Unknown Member Action	A Read-Only field that displays the existence of an Unknown member and whether that member is visible or hidden.
Unknown Member Name	A Read-Only field that displays the caption for the unknown member.

© Copyright, WhereScape Software Ltd All rights reserved.

OLAP Dimension Objects

OLAP Dimension Overview

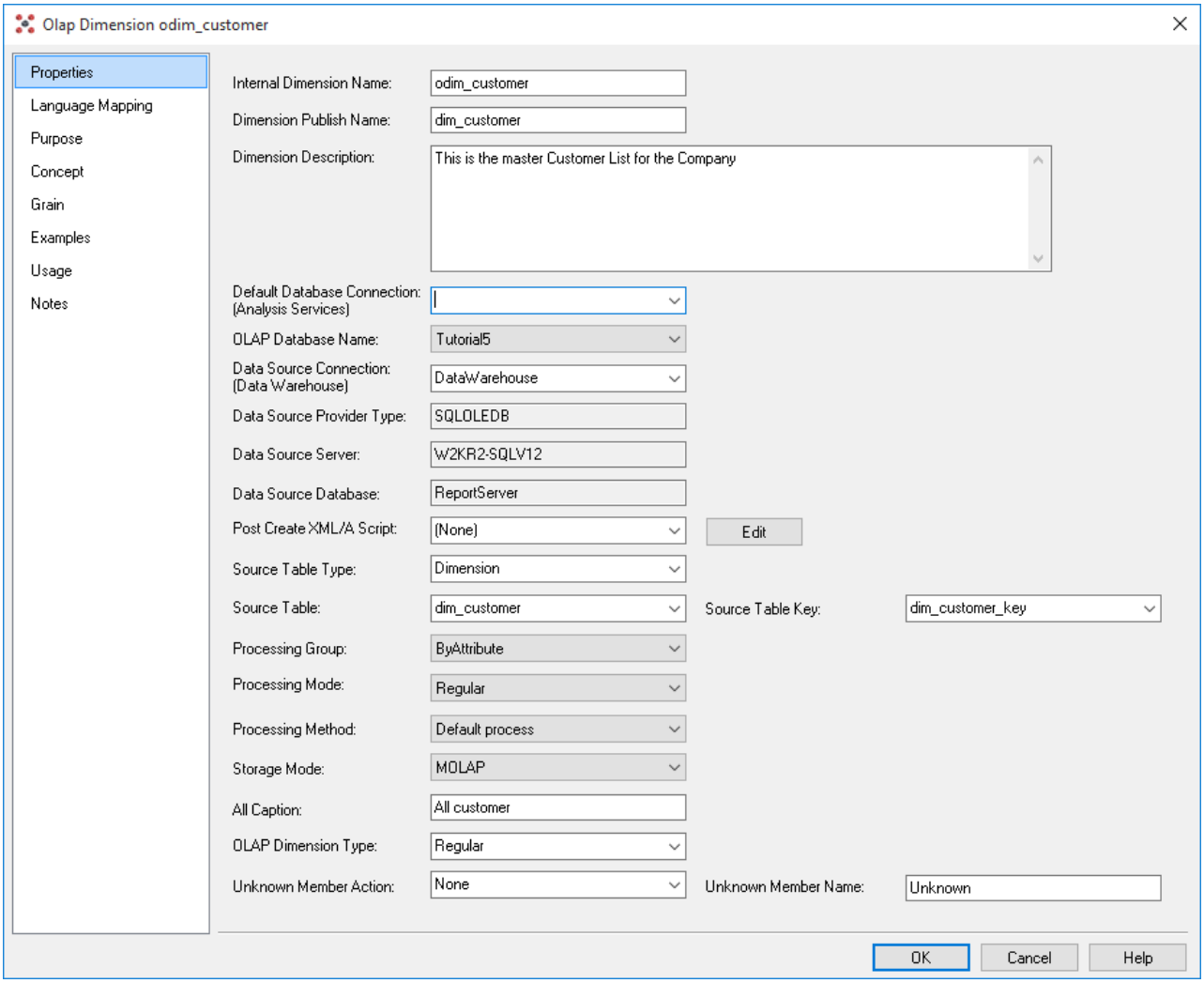
OLAP Dimensions are dimensions that get created in an Analysis Services database.

An OLAP Dimension is a collection of related attributes which can be used to provide information about fact data in one or more cubes. By default, attributes are visible as attribute hierarchies and can be used to understand the fact data in a cube. Attributes can be organized into user-defined hierarchies that provide navigational paths to assist users when browsing the data in a cube.

They are typically created and populated from a relational dimension.

One or more OLAP Dimensions are defined automatically by WhereScape RED when a Fact table is dragged over to create a cube or measure group. WhereScape RED takes the relational dimension tables and related metadata (including hierarchies) defined in the star schemas and create OLAP Dimensions automatically. They can also be defined manually in WhereScape RED.

The properties of an OLAP Cube dimension are shown by right-clicking the **OLAP Dimension** and choosing **Properties**. The following window is shown:



Fields	Description
Internal Dimension Name	Specifies the name of the dimension in WhereScape RED.
Dimension Publish Name	Specifies the name of the dimension as created in Analysis Services.
Dimension	A business description of the OLAP Dimension for use in documentation— this

© Copyright, WhereScape Software Ltd All rights reserved.

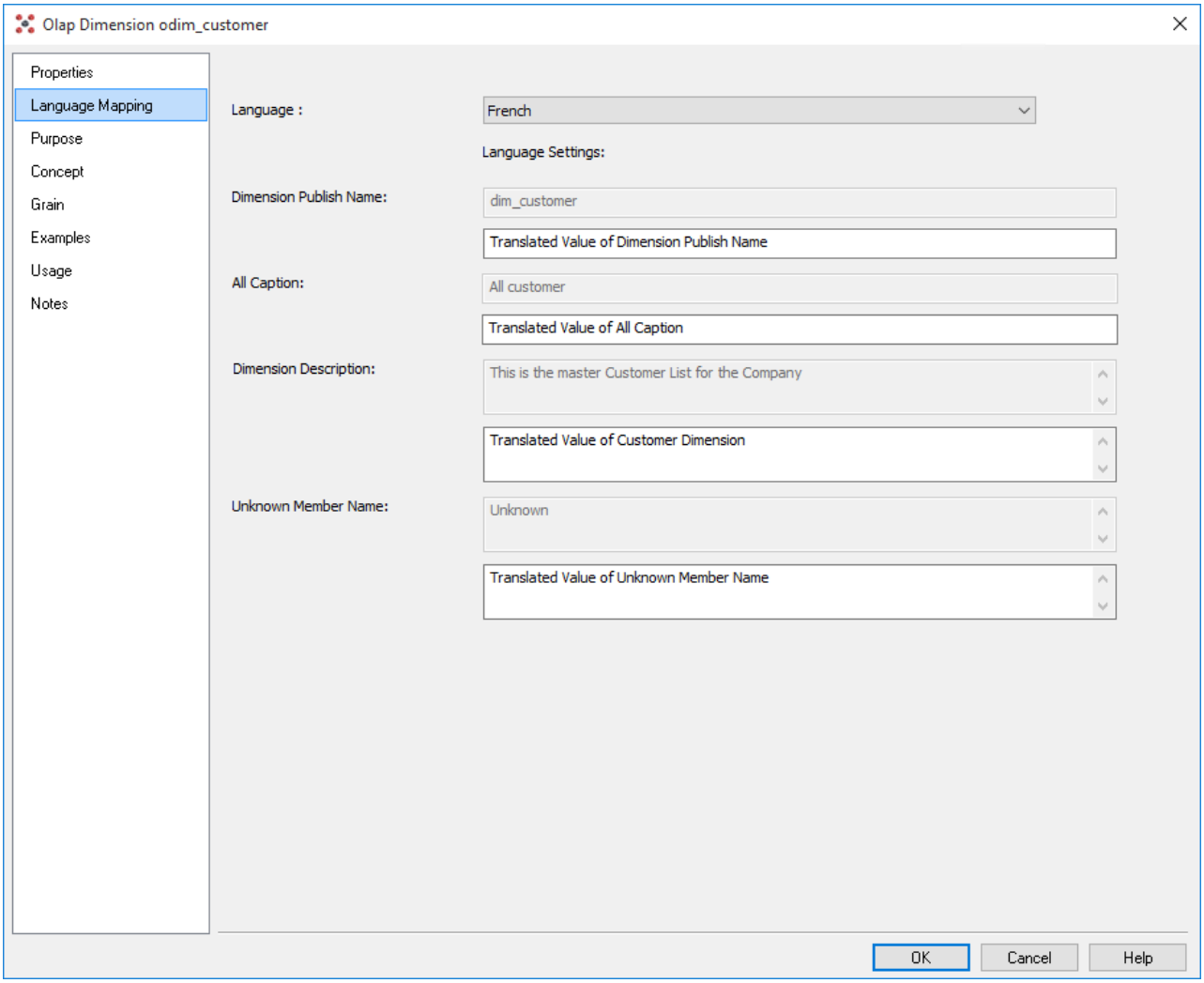
Fields	Description
Description	description also gets created in the analysis services metadata.
Default Database Connection and OLAP Database Name	The WhereScape RED connection that is an OLAP connection to an analysis services server. These fields only need to be populated when the OLAP Dimension needs to be created in Analysis Services separately from a cube. If these fields are blank this dimension can only be created in the same Analysis Services server and database as the related cubes when the cubes get created.
Data Source Connection	Defines the WhereScape RED connection that points to the relational dimensional table(s) used to populate the OLAP Dimension - typically the Data Warehouse connection. When the connection is defined, the following Read-Only fields are populated with the connection information: • Data Source Provider Type • Data Source Server • Data Source Database
Post Create XML/A Script	Specifies XML/A script that is executed after the object is created.
Source Table Type	Specifies the type of table from which the OLAP Dimension was created.
Source Table	Specifies the name of the table from which the OLAP Dimension was created.
Source Table Key	Specifies the key (typically the primary key) that relates the dimension to the fact in the underlying star schema.
Processing Group	Specifies the processing group for processing the dimension. This determines how much data is read into memory during dimension processing at any one time.
Processing Mode	Indicates whether processed data is available after all aggregations have been computed (Regular) or immediately after the data has been loaded (Lazy aggregations).
Processing Method	Indicates which processing method should be used for populating the dimension: • Process Default - Detects the process state of an object, and performs processing necessary to deliver unprocessed or partially processed objects to a fully processed state. • Process Full - Processes an Analysis Services object and all the objects that it contains. When Process Full is executed against an object that has already been processed, Analysis Services drops all data in the object, and then processes the object. This kind of processing is required when a structural change has been made. • Rebuild Data - Processes data only without building aggregations or indexes.
Storage Mode	This field provides two options; MOLAP (Multidimensional OLAP) or ROLAP.(Relational OLAP). This determines how the OLAP cube is processed.
All Caption	Specifies the name of the (All) member. This applies to all hierarchies in the dimension that have an (All) member.
OLAP Dimension Type	Specifies the type of information contained by the dimension. Some client tools can treat the dimension differently based on this information.
Unknown Member Action	Specifies the existence of an Unknown member and whether that member is visible or hidden. Fact data not associated with a member can be associated with the unknown member.
Unknown Member Name	Specifies the caption for the unknown member.

© Copyright, WhereScape Software Ltd All rights reserved.

Language Mapping

The OLAP Dimension Properties screen has a tab called **Language Mapping**.

Select the language from the drop-down list and then enter the translations for the **Dimension Publish Name**, **All Caption**, **Dimension Description** and the **Unknown Member Name**.



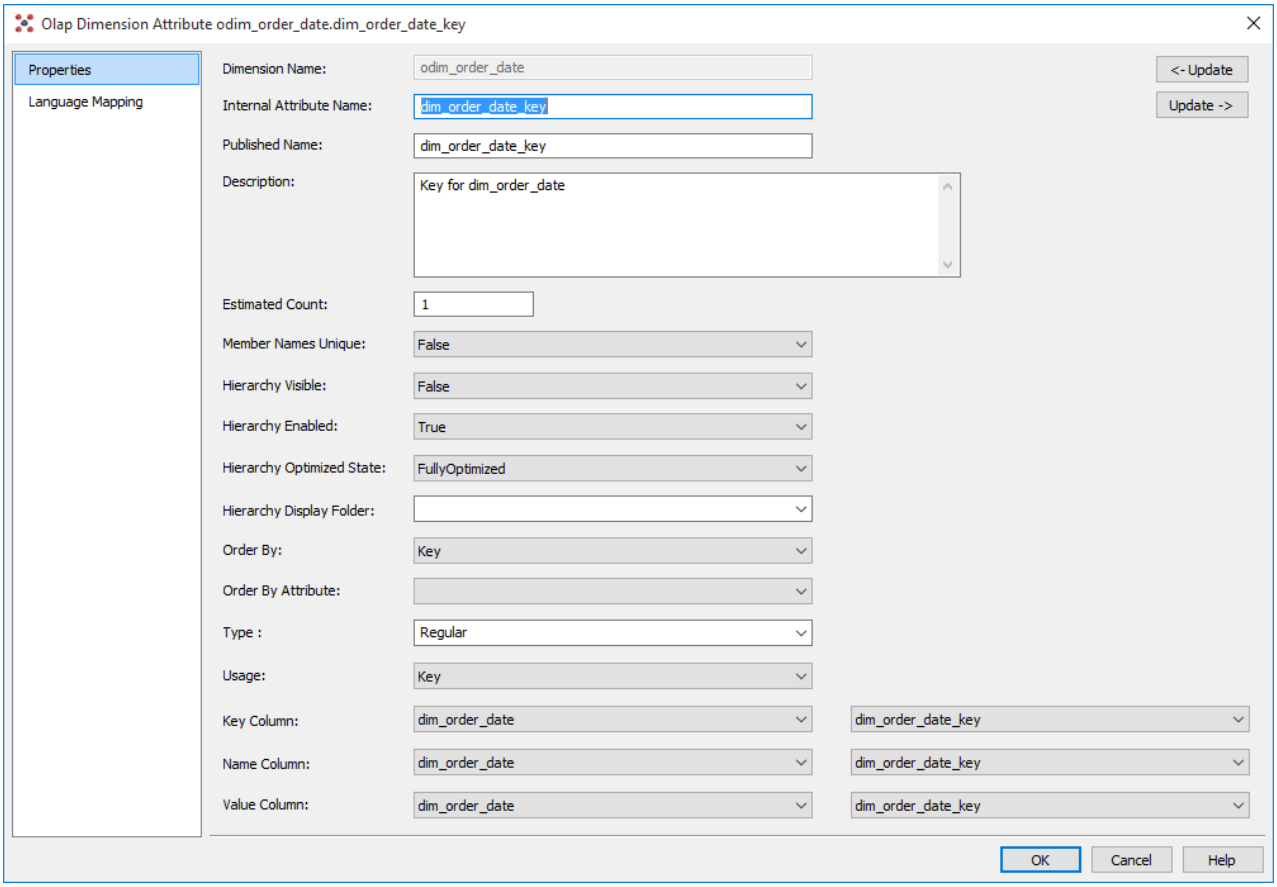
Refer to **Language Options** for more details on how to add languages for translation.

© Copyright, WhereScape Software Ltd All rights reserved.

OLAP Dimension Attributes

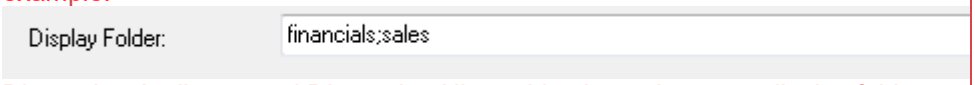
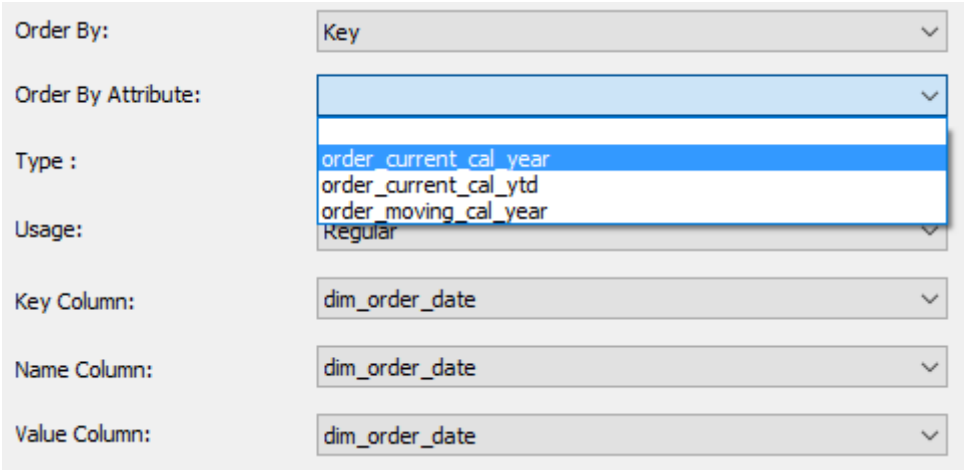
Dimensional attributes contain information about the Dimension object. Attributes are exposed in the cube to provide the ability to navigate and aggregate the data in the cube.
User defined hierarchies can be built over attributes to provide drill paths through the data and to aid aggregation.

The properties of an attribute can be displayed by right-clicking an attribute in the middle pane and choosing **Properties**. The following dialog is displayed:



Fields	Description
Dimension Name	A read only field to display the dimension which the attribute is related to.
Internal Attribute Name	The name of the attribute in WhereScape RED.
Published Name	The name of the attribute created in Analysis Services.
Description	A business name that is stored in WhereScape RED for documentation and stored in the Analysis Services metadata.
Estimated Count	Specifies the number of members in the attribute. This number is either the amount last counted by Analysis Services or a user provided estimate of the member count.
Member Names Unique	Indicates whether member names are unique for this attribute.
Hierarchy Visible	Indicates whether the attribute hierarchy is visible to client applications. Even if the attribute hierarchy is not visible, it can still be used in a user defined hierarchy.
Hierarchy Enabled	Indicates whether an attribute hierarchy is enabled for this attribute. If the attribute hierarchy is not enabled, then the attribute cannot be used in a user defined hierarchy.
Hierarchy Optimized State	Specifies the level of optimization applied to the attribute hierarchy.

© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
Hierarchy Display Folder	Attribute hierarchies can be organized into user-defined folders to view and manage these attributes within the Analysis Services user interface more easily. Enter the display folder name. **Notes:** 1. One object can be in multiple display folders, for example:  2. Dimension Attributes and Dimension Hierarchies have the same display folder drop-down list.
Order by	Specifies the method used to order the members of the attribute.
Order by attribute	Specifies the attribute used to order the members of the attribute hierarchy. If the **Order By** property is set to 'AttributeKey' or 'AttributeName' then **Order By Attribute** cannot be empty. It must be populated with values from attribute relationships. 
Type	Specifies the type of information contained by the attribute.
Usage	Specifies the usage of the attribute.
Key Column	Specifies the details of the binding to the column containing the member key.
Name Column	Specifies the details of the binding to the column containing the member name.
Value Column	Specifies the details of the binding to the column containing the member value.. **Notes:** **Using the Value Column OLAP cube attribute setting for Excel date filtering** In the relevant OLAP Date dimension ensure the OLAP Dimension Type property is set to "Time", then for the Key Attribute of the OLAP Date Dimension (e.g. dim_date_key) set the Value Column property to a date data type column (e.g. calendar_date). Usually it will be useful to set the Name Value property for the Key Attribute to a column containing a textual date format (e.g. dates presented in dd/mm/yy format). After publishing and processing the OLAP cube, use Microsoft Office Excel Pivot Tables to expose date-specific filtering options for the dimension's hierarchies instead of label filtering options.
To View Attributes	To view the list of Attributes, right-click on an OLAP Dimension in the left pane and select **Display Attributes**.
To Add an Attribute	1. Display the attributes of an OLAP Dimension in the middle pane. Display the columns of the dimension in the right pane then drag over a column from the underlying relational dimension into the middle pane.

© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
	2. To add an Attribute, right-click on an OLAP Dimension in the left pane and select **Add Attribute**. This can also be achieved by displaying Attributes in the middle pane, right-clicking and selecting **Add Attribute**. Fill out the dialog box with the relevant details. 3. To create an Attribute that is similar to an existing Attribute, display the Attributes in the middle pane and select **Copy Attribute**. The same dialog box appears as for **Add Attribute** with most of the fields filled in. Notice that the Attribute name has the suffix "- Copy". Change the name in the Attribute Name field and make any other alterations and then click OK.
To Delete an Attribute	Display the Attributes in the middle pane,right-click the Attribute and then select **Delete**.

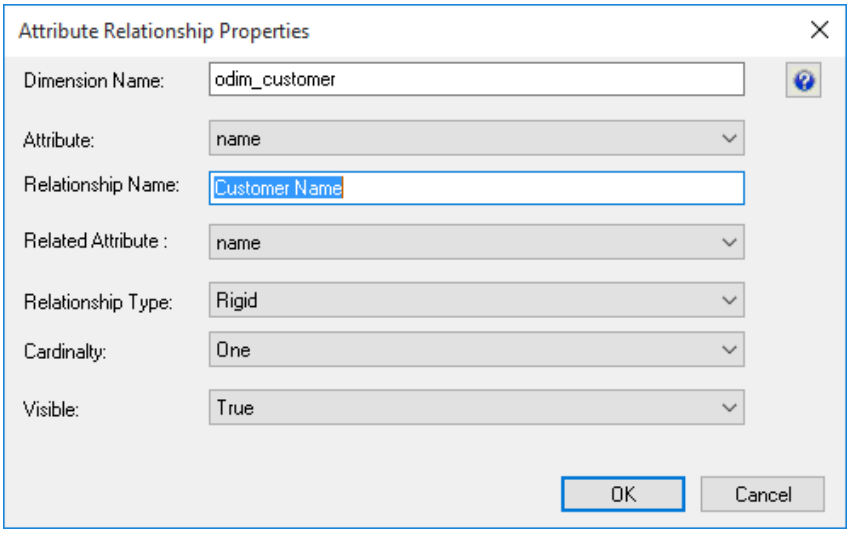
OLAP Dimension Attribute Relationships

Attribute relationships define functional dependencies between attributes. If attribute A has a related attribute B in a hierarchical relationship, then an attribute relationship can be defined that assists Analysis Services to aggregate data.

Note:

Attribute relationships require unique members in order to function correctly—if there are duplicate values participating in an attribute relationship then cube data can be incorrectly aggregated and displayed

To view the **Properties** of an attribute relationship right-click an Attribute Relationship and then select **Properties** to show the window below:



Fields	Description
Dimension Name	A read only field indicating the dimension associated with the attribute relationship.
Attribute	Specifies the attribute on which the attribute relationship is based.
Relationship Name	Specifies the name of the attribute relationship.
Related Attribute	Specifies the name of the related attribute.
Relationship Type	Indicates whether the relationship between attributes can change over time.
Cardinality	Indicates if the related attribute has a many to one or a one to one relationship with this attribute.
Visible	Indicates whether the attribute relationship is visible to the client applications.

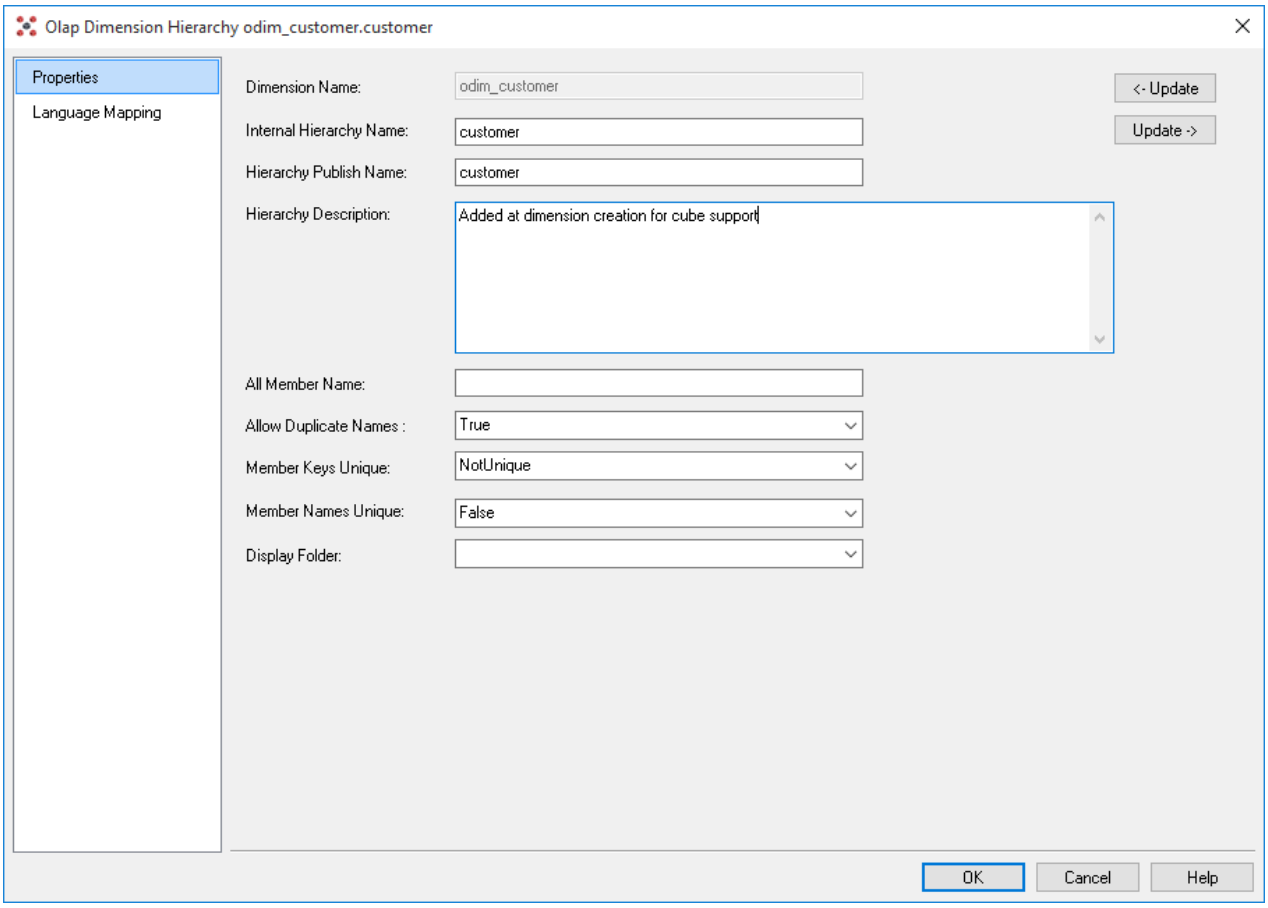
© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
To View Attribute Relationships	To view the list of Attribute Relationships, right-click the OLAP Dimension in the left pane and then select **Display Attribute Relationships**.
To Add an Attribute Relationship	1. Attribute relationships are defined automatically when a User Defined hierarchy is inherited from an underlying relational dimension. 2. To add an Attribute Relationship, right-click the OLAP Dimension in the left pane and then select **Add Attribute Relationship**. Fill out the dialog box with the relevant details.
To Delete an Attribute Relationship	Display the Attribute Relationships in the middle pane, right-click the Attribute Relationship and then select **Delete**.

OLAP Dimension Hierarchies

User defined hierarchies define hierarchical relationships between related dimensional attributes (e.g. Geographical or Time based attributes). These related attributes are defined as levels within the hierarchy.

To view the properties of a user defined Hierarchy, right-click the **Hierarchy** and then select **Properties**.



Fields	Description
Dimension Name	A read only field indicating the dimension associated with the attribute relationship.
Internal Hierarchy Name	The name of the hierarchy within WhereScape RED.
Hierarchy Publish Name	The name of the hierarchy as created within Analysis Services.

© Copyright, WhereScape Software Ltd All rights reserved.

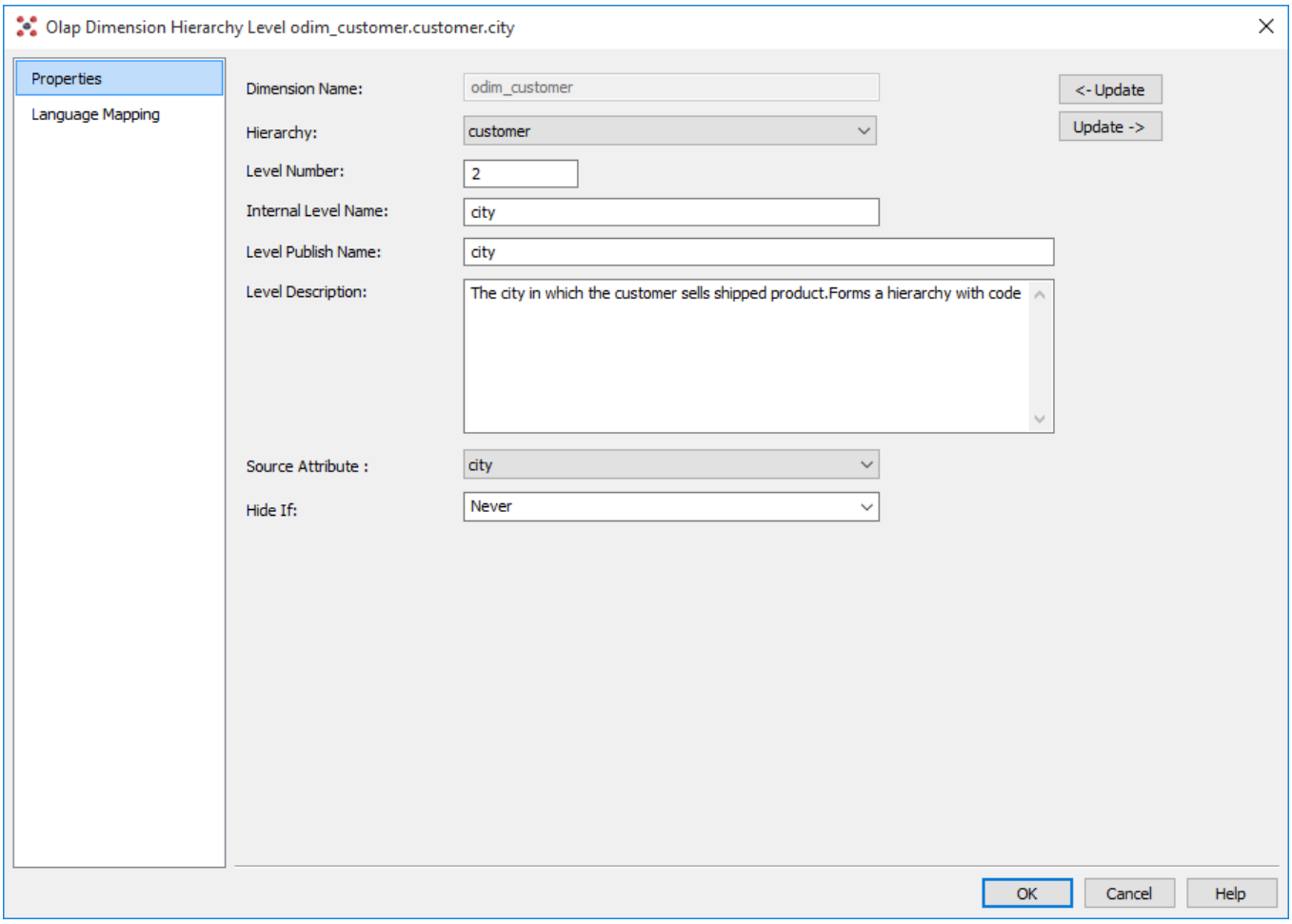
Fields	Description
Hierarchy Description	A business name that is stored in WhereScape RED for documentation and stored in the Analysis Services metadata.
All Member Name	Specifies the name of the member in the All level
Allow duplicate Names	Indicates whether the members under a common parent can have the same name.
Member Keys Unique	Indicates whether member keys are unique for this hierarchy. . **Notes:** If you are using a version of Analysis Services earlier than service pack 2, the only value allowed for **Member Keys Unique** is 'NotUnique'. If the value 'Unique' is used, Analysis Services will return an error and the cube will not be created.
Member Names Unique	Indicates whether member names are unique for this hierarchy.
Display Folder	Dimension hierarchies can be organized into user-defined folders to view and manage these attributes within the Analysis Services user interface more easily. Enter the display folder name. **Notes:** 1. One object can be in multiple display folders, for example: Display Folder: financials;sales 2. Dimension Attributes and Dimension Hierarchies have the same display folder drop-down list.
To view User Defined Hierarchies	To view the list of User Defined Hierarchies, right-click on an OLAP Dimension in the left pane and select **Display Hierarchies**.
To Add a User Defined Hierarchy	To add a User Defined Hierarchy, right-click on an OLAP Dimension in the left pane and select **Add Hierarchy**. Fill out the dialog box with the relevant details.
To Delete a User Defined Hierarchy	Display the User Defined Hierarchies in the middle pane, select the User Defined Hierarchy to delete, right-click and select **Delete**.

OLAP Dimension User Defined Hierarchy Levels

The levels specify the drill path over a set of related attributes. The classic hierarchy levels are Year, Month, Date in a Calendar based hierarchy in the date dimension.

To view the Properties of a user defined Hierarchy Level, right-click a user defined Hierarchy Level and then select **Properties**.

© Copyright, WhereScape Software Ltd All rights reserved.



Fields	Description
Dimension Name	A Read-Only field indicating the dimension associated with the attribute relationship.
Hierarchy	The User Defined Hierarchy that contains the level.
Level Number	The number of levels from the top most hierarchy level. These level numbers must start at 1 for the top level and provide continuous numbering to the bottom level.
Internal Level Name	The name of the Level within WhereScape RED.
Level Publish Name	The name of the Level as created within Analysis Services.
Level Description	A business name that is stored in WhereScape RED for documentation and stored in the Analysis Services metadata.
Source Attribute	Specifies the source attribute on which the level is based.
Hide If	Specifies which members are hidden. This property supports ragged hierarchies containing logical gaps between members.
To View User Defined Hierarchy Levels	To view the list of User Defined Hierarchy Levels, right-click the OLAP Dimension in the left pane and then select **Display Hierarchy Levels**.
To Add a User Defined Hierarchy	1. To add a User Defined Hierarchy level, right-click the OLAP Dimension in the left pane and then select **Add Hierarchy Level**. Fill out the dialog box with the relevant details. 2. Alternatively right-click the User Defined Hierarchy in the middle pane and then select **Add Hierarchy Level**. Fill out the dialog box with the relevant details.

© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
To Delete a User Defined Hierarchy Level	Display the User Defined Hierarchy Levels in the middle pane, right-click the User Defined Hierarchy Level and then select **Delete**.

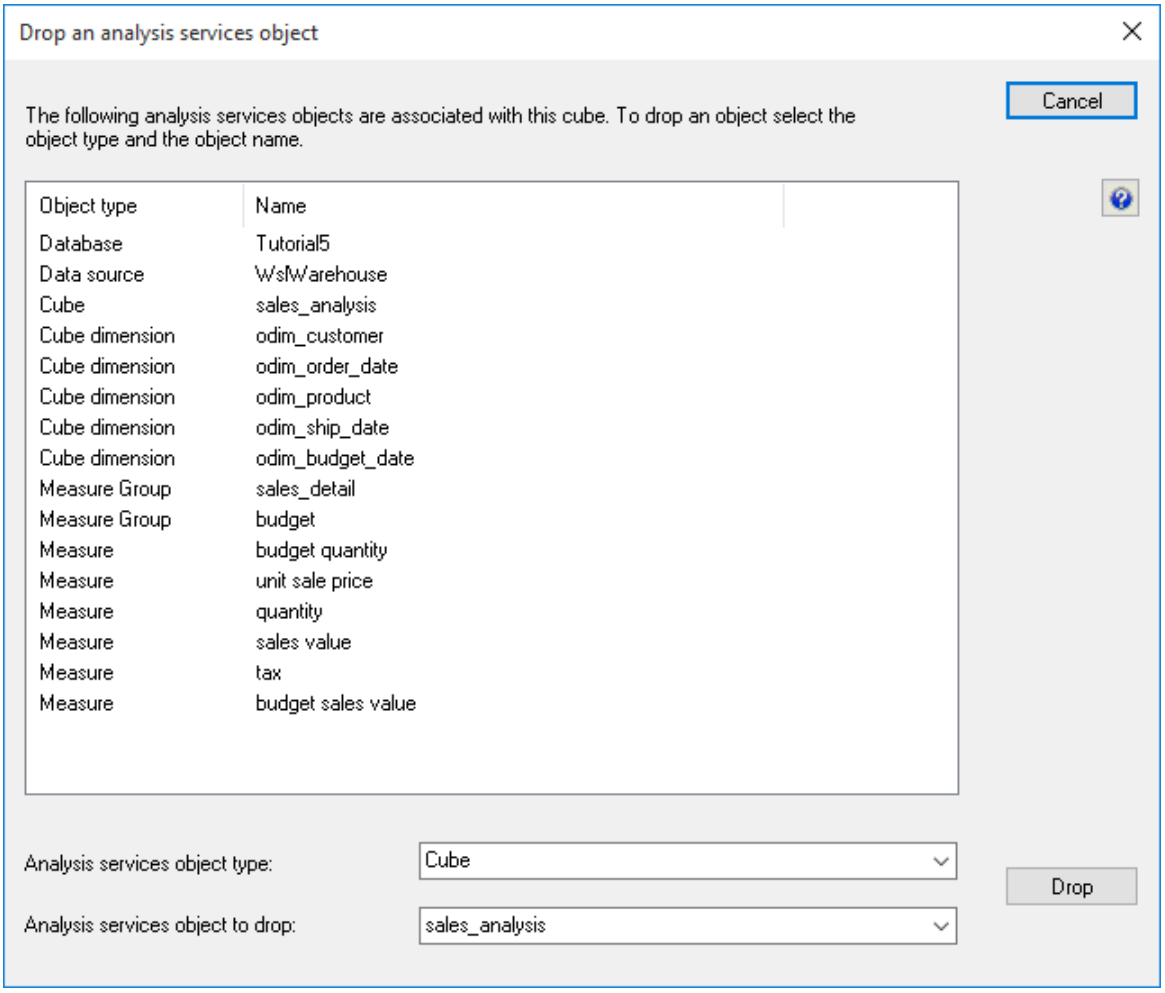
OLAP Changing OLAP Cubes

An understanding of the dependency of objects within Analysis Services is the key to figuring out what needs to be dropped or recreated in a cube database using WhereScape RED.

Changes to the underlying relational star schema can cause cube processing to fail as the star schema is frozen in the Data Source View (DSV) of the cube database. Minor changes such as the addition of table columns, or altered data types (e.g. changing a char to varchar) does not break the cube, but renaming a source table or column that is used as a source to the cube invalidates the DSV and cause processing to fail.

The solution to this issue is to drop and recreate the Cube database from RED to recreate the DSV or manually update the DSV using Microsoft BIDS.

If an object needs to be dropped and recreated in RED, then these are two separate actions. For example to drop the OLAP database, right-click an OLAP cube within that database in RED and select **Drop Analysis Services Object**, and then either double-click an object in the list or use the drop-down boxes in the Drop Analysis Services Object window to choose the object to drop, and click **Drop**. This will drop the object from Analysis Services.



A Create action on an Analysis Services object in RED will be different depending on whether or not the object already exists in Analysis Services:

- If the object does not already exist in Analysis Services then RED will create the object (and any related objects e.g. OLAP database and DSV).

© Copyright, WhereScape Software Ltd All rights reserved.

- If the object does already exist in Analysis Services then RED will try to detect any changes or additional features that need to be added to the object and add or alter the existing Analysis Services object.

Some objects need to be dropped and recreated in order to be changed (e.g. dimension structures), and some only need to be recreated (e.g. calculations).

Changes to cube runtime objects do not require the cube database to be dropped. For example a new or changed definition of a calculation or KPI will not require the cube to be dropped and recreated (so data is retained). Recreating the cube, updates the definition of these runtime objects and immediately becomes available to cube users.

A brief summary of the hierarchy of objects and the remedial action is shown below:

Cube Object	Change	Action
Data Source	This changes the source database connection. It is defined in the Data Warehouse connection in RED.	OLAP database needs to be dropped and recreated.
Data Source View (underlying relational star)	The DSV reflects the design of the relational star. Therefore, the DSV would need to be updated if any changes are made to tables or views that are used to build OLAP objects.	Changes to the underlying relational star that affect an existing OLAP Object requires that the OLAP Database is dropped and recreated to regenerate the DSV.
OLAP Dimension	The addition or deletion of attributes or hierarchies to an existing OLAP dimension.	The OLAP dimension plus any OLAP cubes associated with the dimension need to be dropped and recreated.
OLAP Cube Measure Group	Delete or Add a Measure Group based on a fact that already exists in the DSV.	Recreate the cube in RED and reprocess.
OLAP Cube Measure Group	Add a Measure Group based on a fact that does not exist in the DSV.	Recreate the OLAP cube database and reprocess.
OLAP Cube Measures	Delete or Add measures based on columns that already exist in the DSV.	Recreate the cube in RED and reprocess.
OLAP Cube Measures	Add measures that are based on new columns that do not exist in the DSV.	Recreate the OLAP cube database and reprocess.
OLAP Cube Calculations, KPIs, Actions	Add, change or delete definition on the cube.	Recreate the cube in RED (a reprocess is not necessary because just the calculation definition is stored in the cube—the result is calculated at query time).

© Copyright, WhereScape Software Ltd All rights reserved.

OLAP Retrofitting an OLAP Object

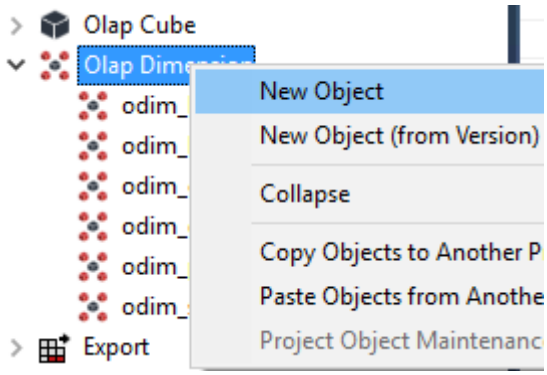
WhereScape RED provides the functionality for retrofitting OLAP cubes from Analysis Services.

Note:

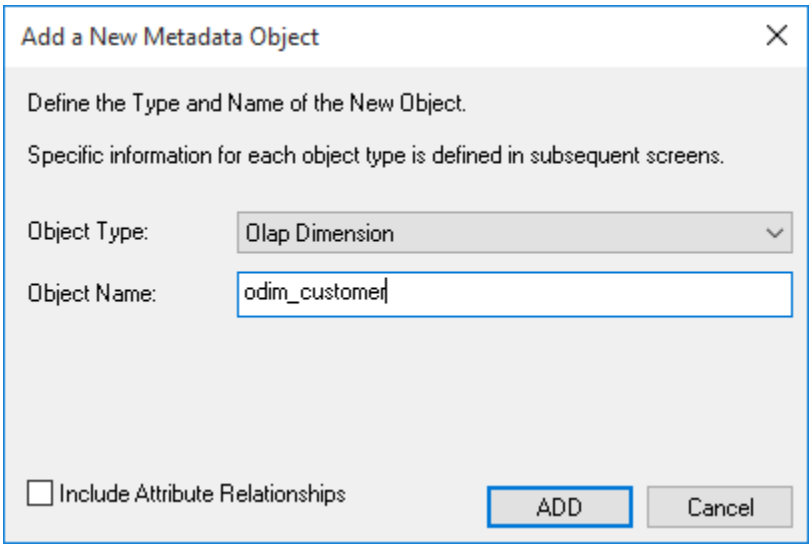
Before you can retrofit an OLAP cube, you must first retrofit any and all of the OLAP dimensions used by the OLAP cube.

The process to retrofit an OLAP dimension is as follows:

1. Right-click the **OLAP Dimension** object heading in the left pane and select **New Object**.

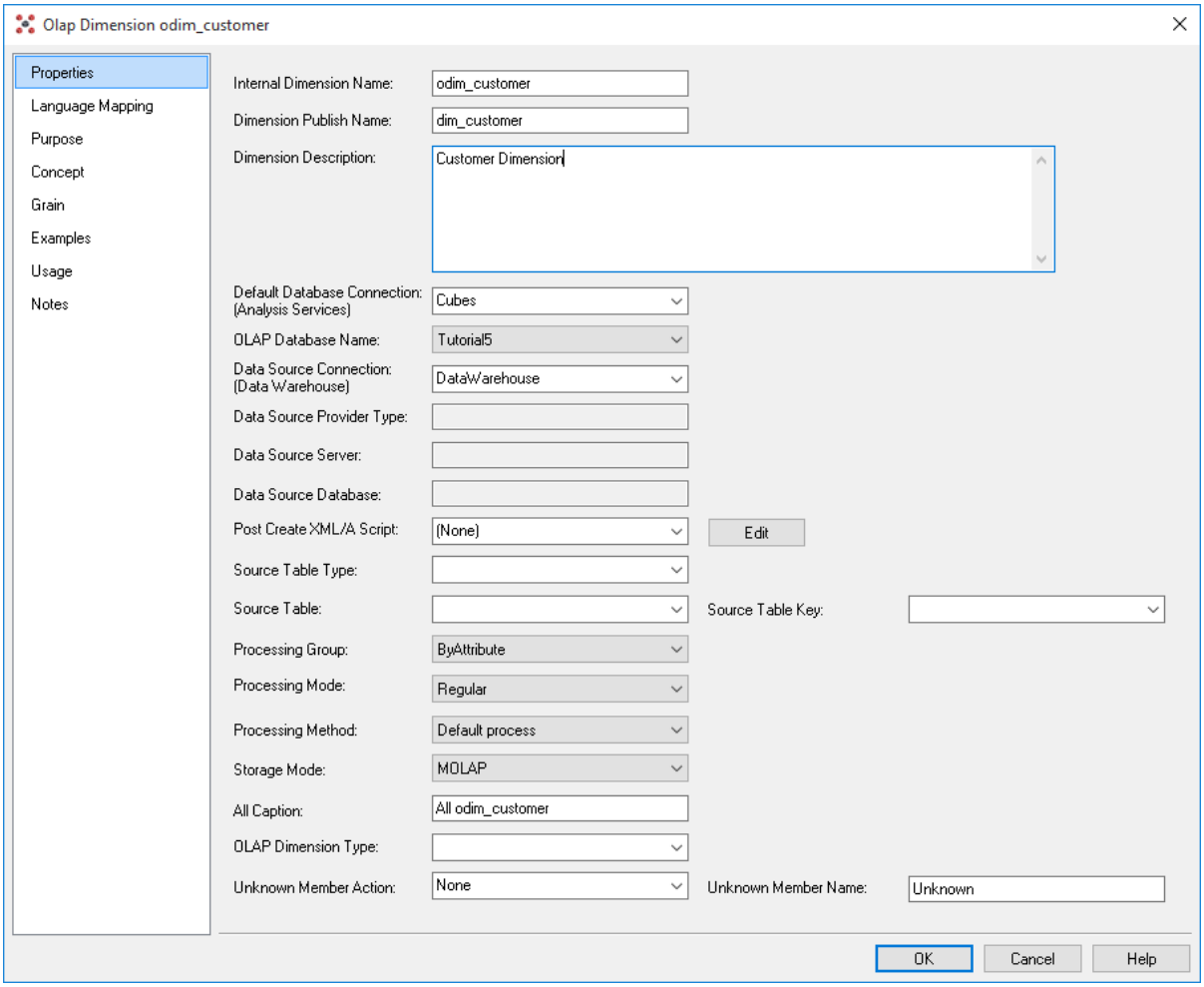


2. Enter any name for the object name and click **ADD**.



© Copyright, WhereScape Software Ltd All rights reserved.

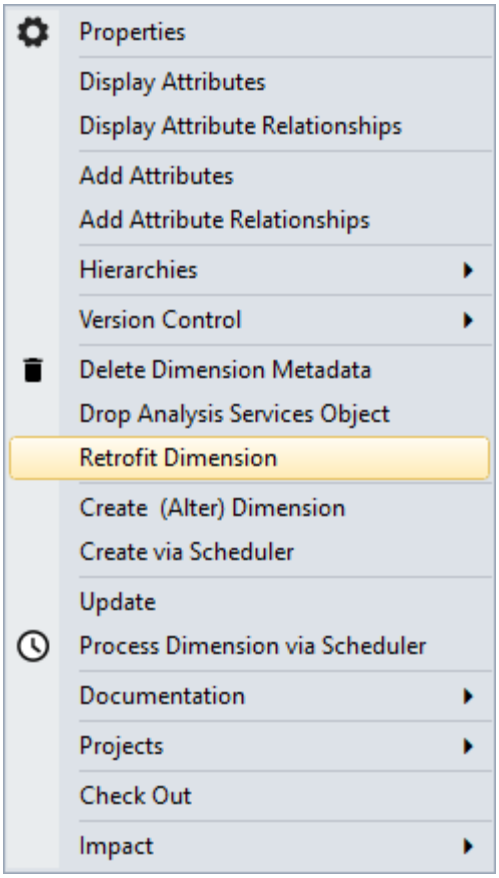
3. Fill the Properties window fields described below:



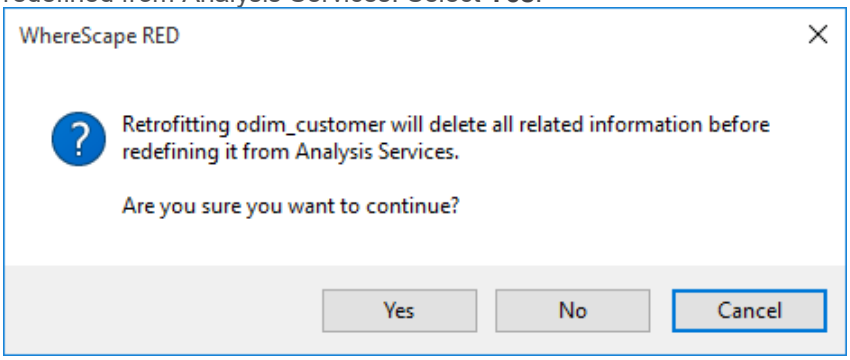
Options	Description
Internal Dimension Name	The name that matches exactly the name of the OLAP Dimension in Analysis Services that you want to retrofit.
Dimension publish name	The name for the dimension.

© Copyright, WhereScape Software Ltd All rights reserved.

4. The OLAP Dimension can now be retrofitted by right-clicking the object in the left pane and choosing **Retrofit Dimension**.

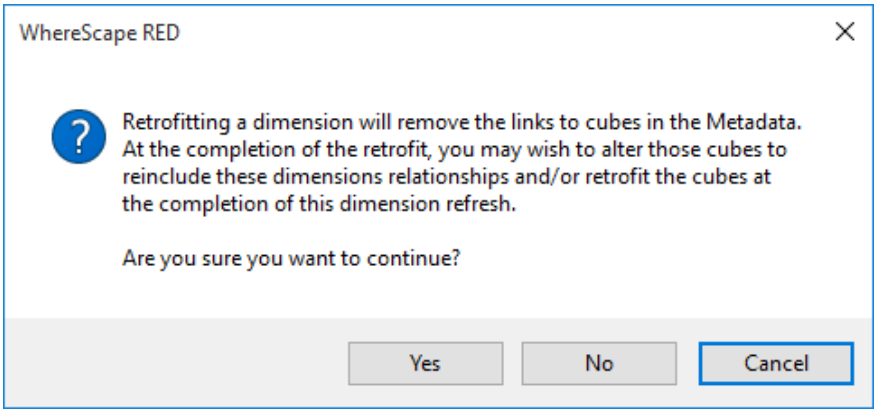


5. Two warning prompts appear. The first warns that the existing information is deleted before being redefined from Analysis Services. Select **Yes**.



© Copyright, WhereScape Software Ltd All rights reserved.

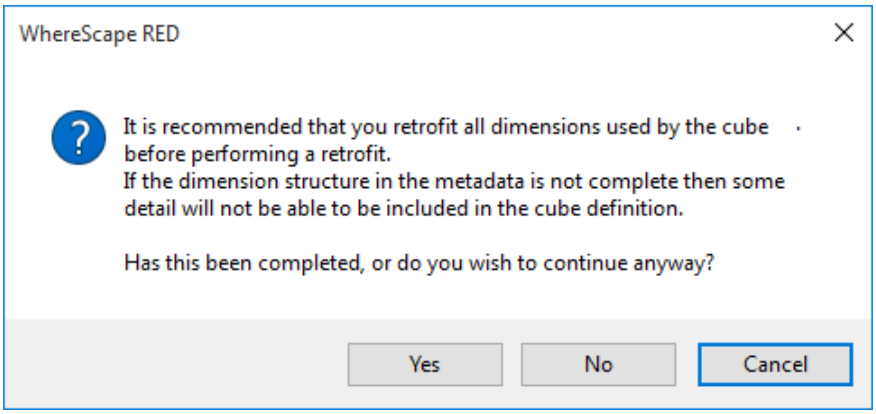
6. The second prompt warns that retrofitting a dimension will remove the links to cubes in the Metadata. Select **Yes**.



7. The results of the retrofit are displayed in the results panel at the bottom of the screen. If it was successful, the new object should now have all the attributes, hierarchies and hierarchy levels (as well as all their properties) as set in Analysis Services.

Once all the dimensions have been retrofitted, you can retrofit the **OLAP Cube**. Follow steps 1-6 above to retrofit the OLAP Cube as for the OLAP dimensions.

The final dialog is a reminder to only retrofit an OLAP cube once the dimensions used by the cube have been retrofitted. If all relevant dimensions have been retrofitted, select **Yes**.



Once again, the results of the retrofit are displayed in the results panel at the bottom of the screen.

© Copyright, WhereScape Software Ltd All rights reserved.

Tabular Models

WhereScape RED MSAS's Tabular Mode functionality is enabled for SQL Server, Oracle and Teradata repositories, available only for Dimension, Fact and EDW3NF objects.

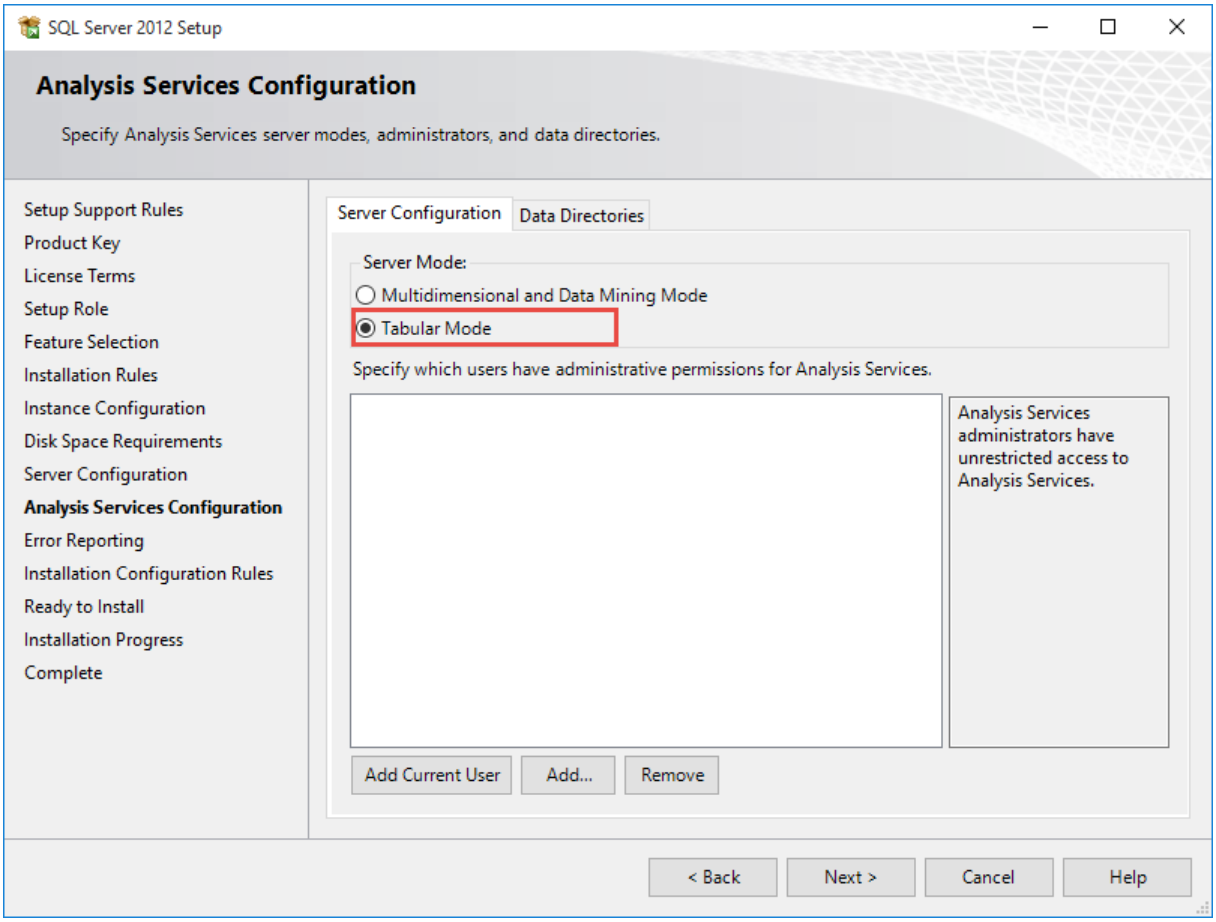
To use this RED functionality, users need to ensure they have a Microsoft Analysis Services Server running in Tabular mode available.

The following sections provide an overview of the Tabular functionality in RED, detailing the following:

- Ensuring the installation of a Microsoft Analysis Services Server is running in Tabular mode
- The creation of a new Tabular connection with associated tabular targets
- Setup of an MSAS connection string in the DataWarehouse connection
- Creating Fact, Dimension or EDW3NF tables by Dragging and Dropping, then choosing a Tabular Target Location.
- Querying the results via Excel

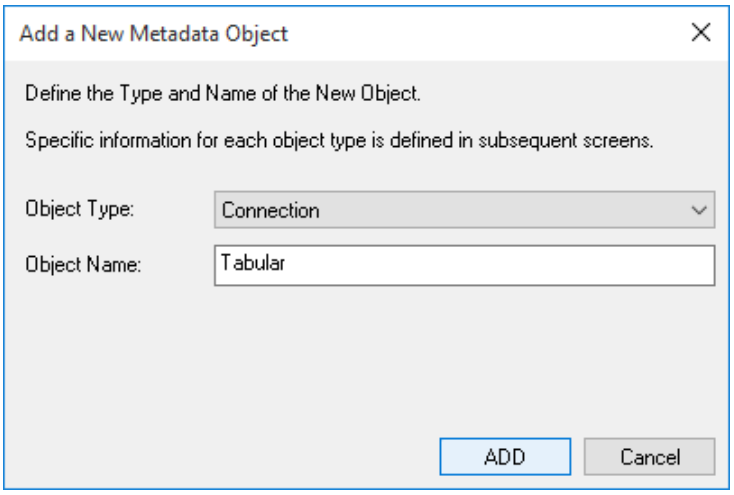
Defining the Data Source and Targets for Tabular Models

1. To use the MSAS Tabular mode functionality in RED, start by following your SQL Server setup's Installation to "New SQL Server stand-alone installation or add features to an existing installation" option to ensure that your Analysis Server Configuration has been installed for **Tabular Mode**.



2. In RED, create a new Tabular **Connection**.
   - Give your connection a relevant name for your system. In this example, we have named the connection "Tabular".
   - Click **Add**.

© Copyright, WhereScape Software Ltd All rights reserved.

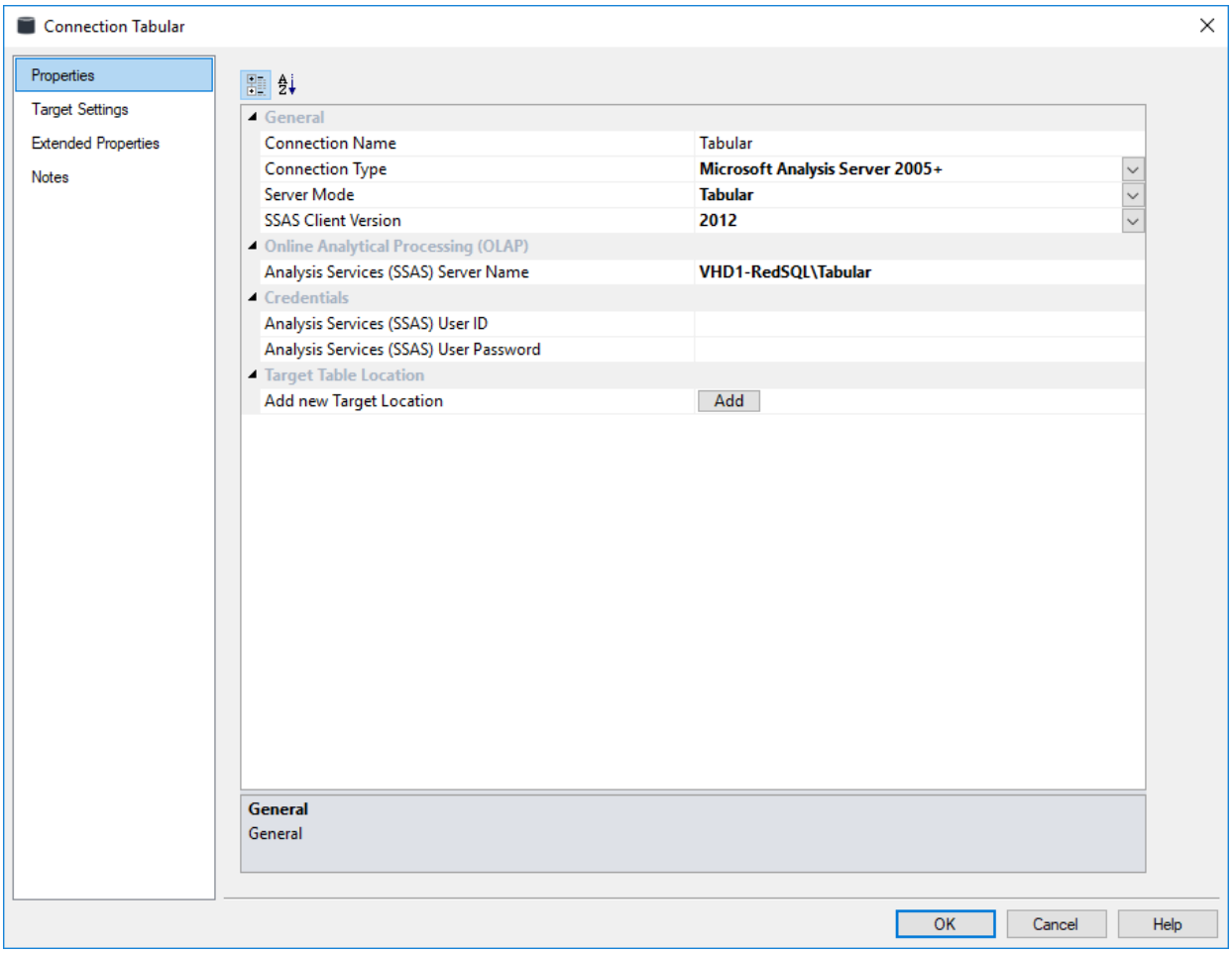


3. Select "Microsoft Analysis Server 2005+" for your **Connection Type**.
   - Select "Tabular" for your **Server Mode**.
   - Select the **SSAS Client Version** to connect for your SSAS database.

Note:

It is recommended that the client version matches the database version.

   - For the **Analysis Services (SSAS) Server Name**, Type in the same server name used when installing Microsoft Server Analysis Services in Tabular Mode



Notes:

- If the required SQL Server Analysis Management Objects (AMO) are missing, see the following article for more information: **https://msdn.microsoft.com/en-**

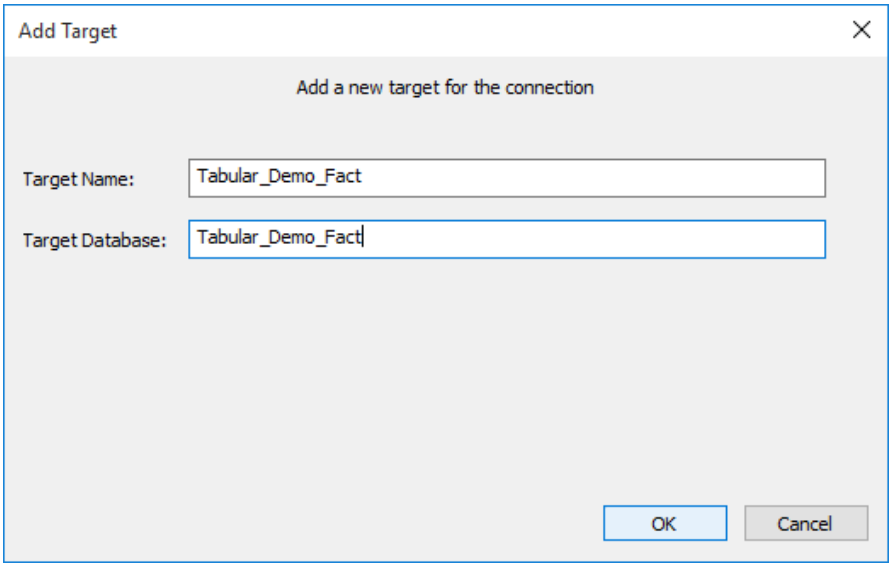
© Copyright, WhereScape Software Ltd All rights reserved.

Notes:

us/library/dn141152.aspx.

- You may need to specify the port number of the Analysis Services instance. To find your port number, follow the procedure documented in this Microsoft article: **https://support.microsoft.com/en-us/kb/2466860**.
  An example of your Analysis Server (SSAS) Server Name using the port number in RED would be: VH1D-REDSQL:49449\TABULAR

4. Create your Tabular targets by clicking the **Add** button and entering a name for the Target and the Target Database.
   - The **Target Name** is the relevant Tabular Database's name displayed in RED.
   - The **Target Database** is the relevant Tabular Database's name displayed in Analysis Services.

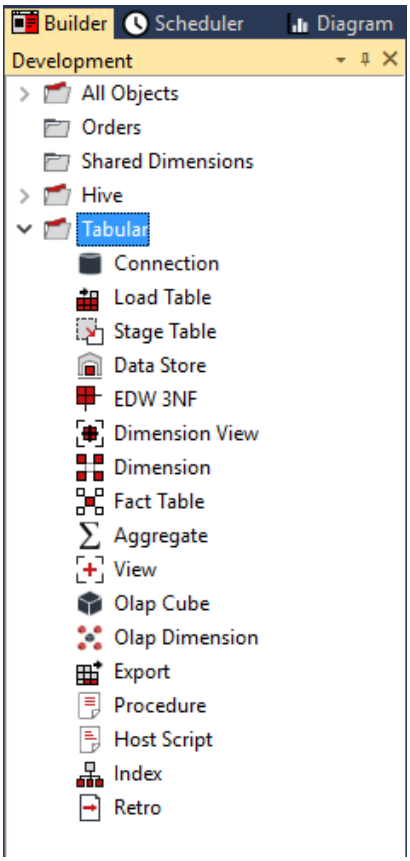


5. For the ease of exploring the new Tabular functionality in RED and differentiate it from the existent database tables, we have created another Project and named it "Tabular".

Note:

To show unused object types in the browser pane enable **Project Properties/Unused Object Types in Object Browser**.

© Copyright, WhereScape Software Ltd All rights reserved.

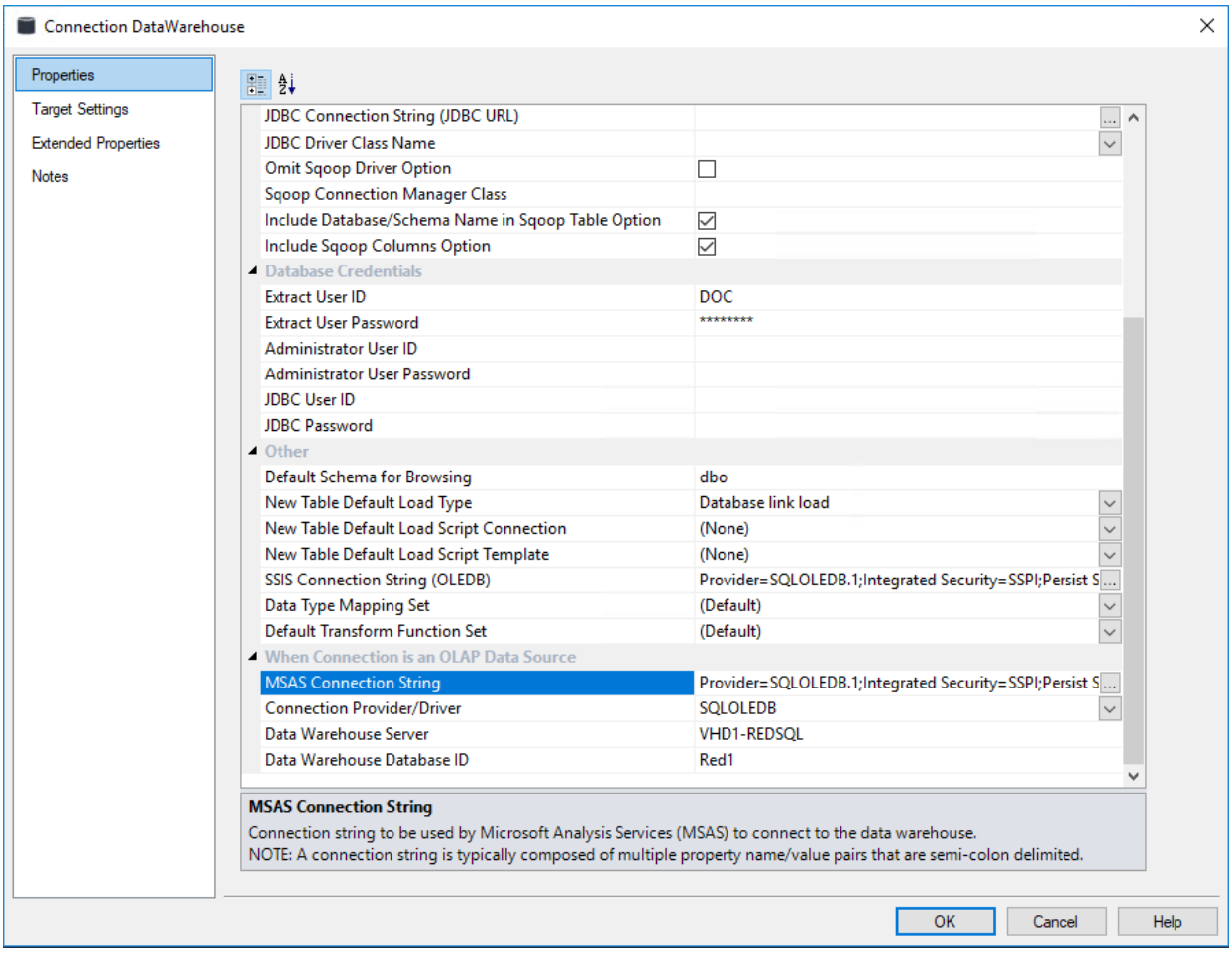


6. Double click your **Datawarehouse** connection and ensure you have set your MSAS Connection String and other relevant Analysis Server required fields.

Note:

If you have already set the MSAS Connection String for use with Multidimensional (OLAP) Cubes, you should not need to change it.
For more information about setting up this connection string, refer to **Defining the Data Source for OLAP Cube** for details.

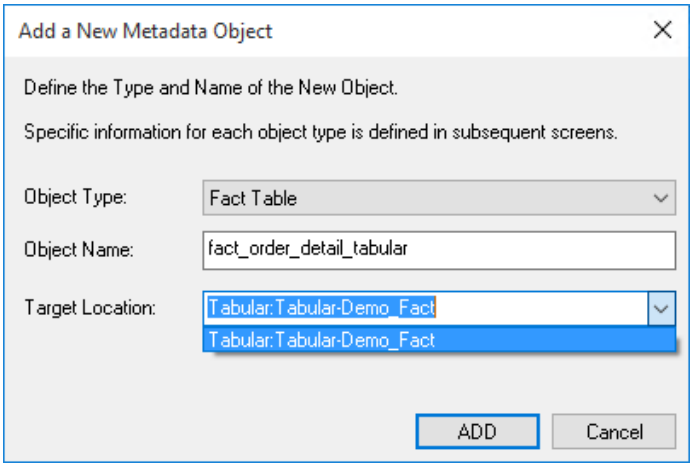
© Copyright, WhereScape Software Ltd All rights reserved.



Defining a Tabular Model

1. Browse your **Data Warehouse** connection.
2. Within your Tabular Project, click the **Fact Table** object group on the left pane to set your work pane to be your Fact Tables.
3. From the right pane, drag a **Fact Table** into the middle pane, rename it with a tabular or relevant reference and select the Tabular target you want to place the table in.

   We have renamed this table to "fact_order_detail_tabular" and selected the "Tabular_Demo_Fact target" to place it in.

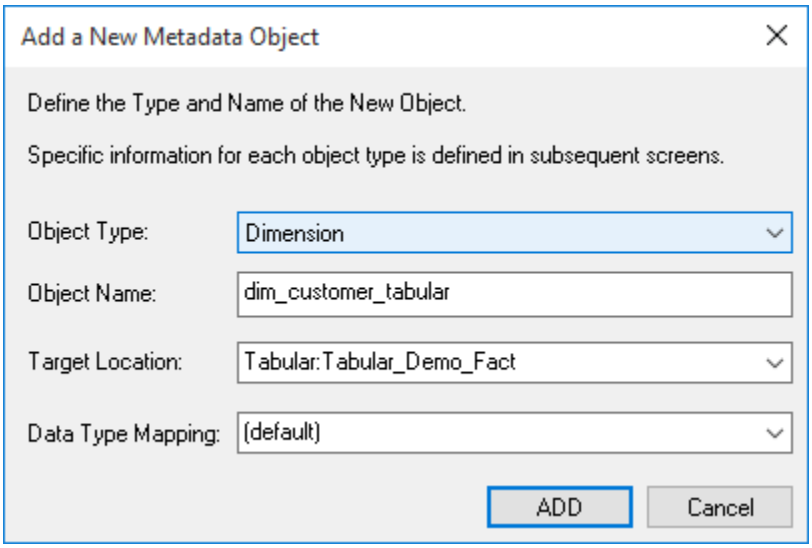


4. Click **OK** in the Fact Table's Properties screen.
5. Right click on the table **fact_order_detail_tabular** from the left pane and select **Create(Recreate)**.

© Copyright, WhereScape Software Ltd All rights reserved.

6. After the table has been recreated, right click again and select **Update** to update the table.
7. Browse your **Data Warehouse** connection again, set your work pane to be Dimension tables and drag your Customer Dimension from the right pane into the middle pane Dimension group.
8. Change the **Object Type** from **Dimension View** to **Dimension** and give the table a relevant name.

   Select a tabular target location to place the table.



9. Click **OK** in the Properties screen.
10. Right click the dim_customer_tabular table from the left pane and select **Create(Recreate)**.
11. After the table has been recreated, right-click again and select **Update** to update the table.
12. Repeat steps 7 to 11 for all other dimension tables joined to your tabular fact table.
13. Right-click the **fact_order_detail_tabular** and select **Query Via Excel** to view the table in Excel.

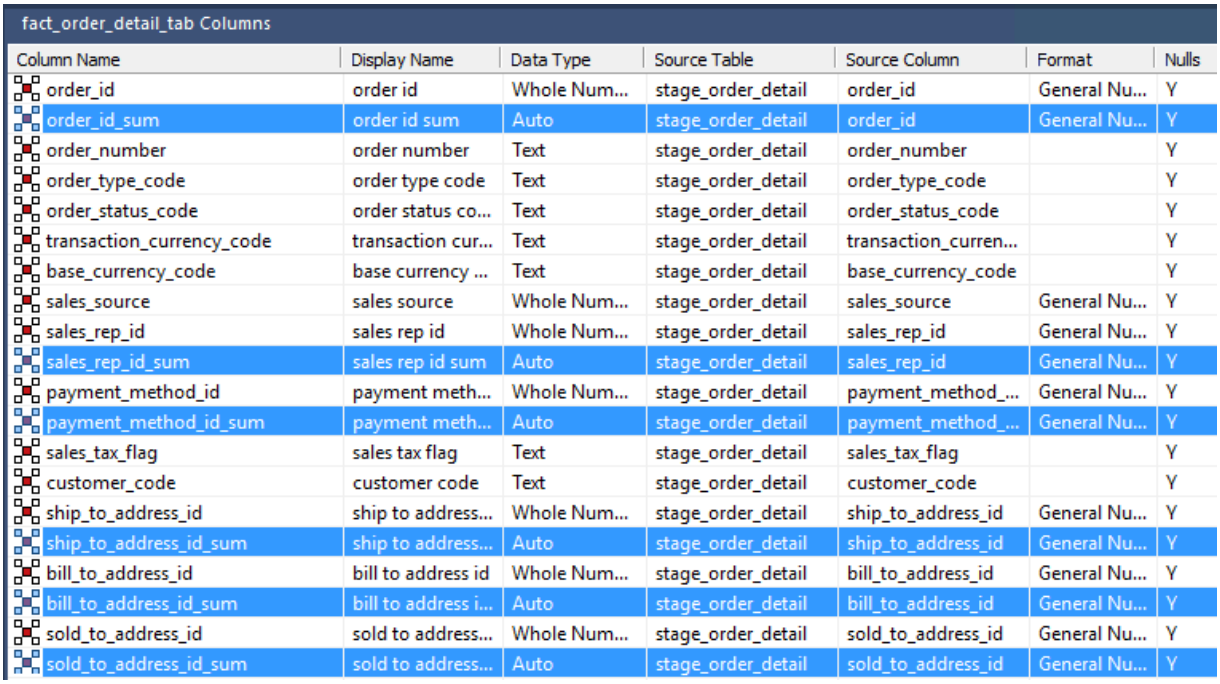
Note:

RED operations such as Validate Against Database, Generate Statistics, Table Row Count and Regenerate Indexes are currently not supported with Tabular mode.

© Copyright, WhereScape Software Ltd All rights reserved.

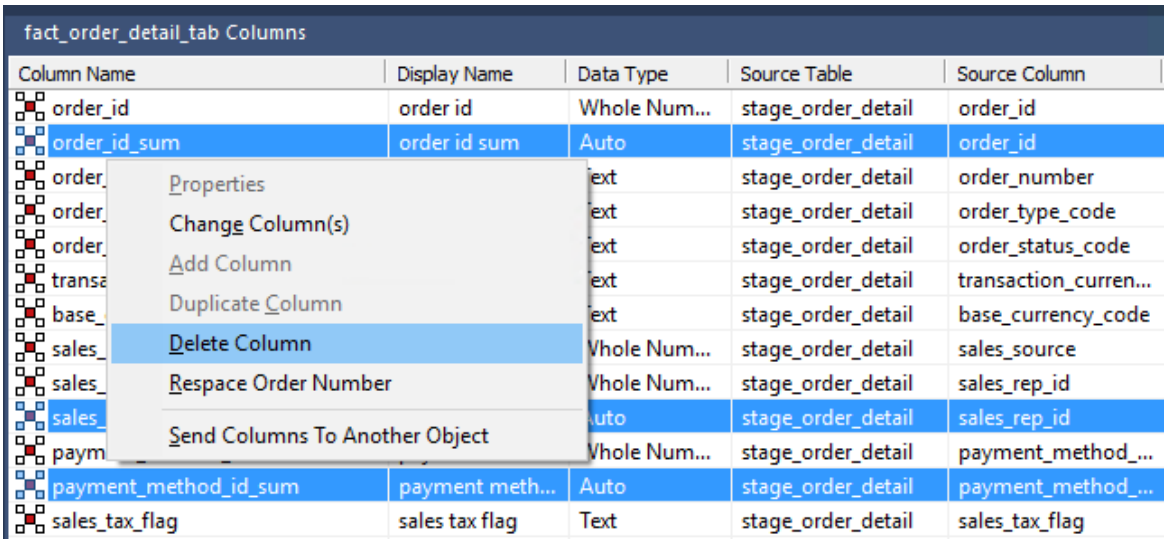
Tabular Model Measures

1. When a new Tabular fact table is created, all numeric columns that are market as additive (apart from key columns) will have auto-created measure totals.


fact_order_detail_tab Columns

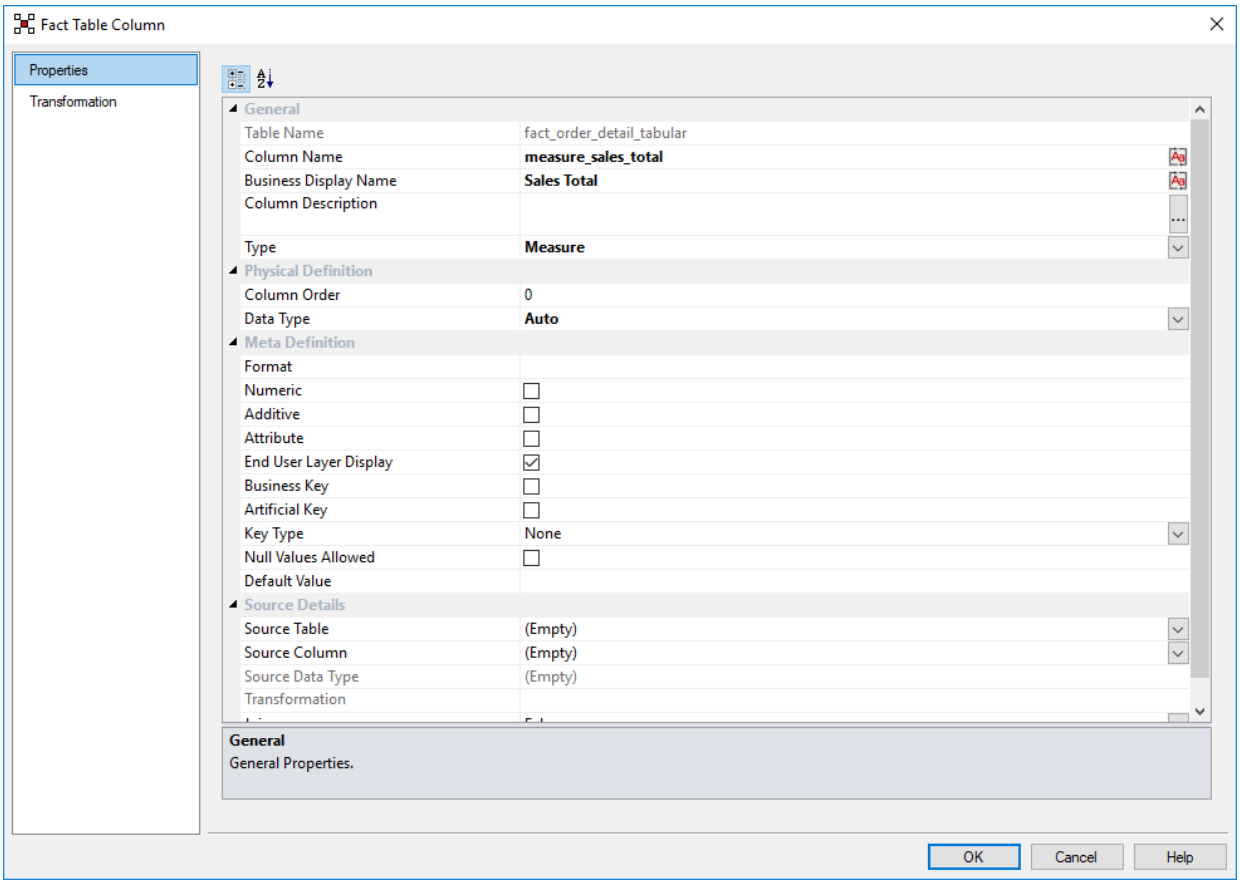
Column Name	Display Name	Data Type	Source Table	Source Column	Format	Nulls
order_id	order id	Whole Num...	stage_order_detail	order_id	General Nu...	Y
order_id_sum	order id sum	Auto	stage_order_detail	order_id	General Nu...	Y
order_number	order number	Text	stage_order_detail	order_number		Y
order_type_code	order type code	Text	stage_order_detail	order_type_code		Y
order_status_code	order status co...	Text	stage_order_detail	order_status_code		Y
transaction_currency_code	transaction cur...	Text	stage_order_detail	transaction_curren...		Y
base_currency_code	base currency ...	Text	stage_order_detail	base_currency_code		Y
sales_source	sales source	Whole Num...	stage_order_detail	sales_source	General Nu...	Y
sales_rep_id	sales rep id	Whole Num...	stage_order_detail	sales_rep_id	General Nu...	Y
sales_rep_id_sum	sales rep id sum	Auto	stage_order_detail	sales_rep_id	General Nu...	Y
payment_method_id	payment meth...	Whole Num...	stage_order_detail	payment_method_...	General Nu...	Y
payment_method_id_sum	payment meth...	Auto	stage_order_detail	payment_method_...	General Nu...	Y
sales_tax_flag	sales tax flag	Text	stage_order_detail	sales_tax_flag		Y
customer_code	customer code	Text	stage_order_detail	customer_code		Y
ship_to_address_id	ship to address...	Whole Num...	stage_order_detail	ship_to_address_id	General Nu...	Y
ship_to_address_id_sum	ship to address...	Auto	stage_order_detail	ship_to_address_id	General Nu...	Y
bill_to_address_id	bill to address id	Whole Num...	stage_order_detail	bill_to_address_id	General Nu...	Y
bill_to_address_id_sum	bill to address i...	Auto	stage_order_detail	bill_to_address_id	General Nu...	Y
sold_to_address_id	sold to address...	Whole Num...	stage_order_detail	sold_to_address_id	General Nu...	Y
sold_to_address_id_sum	sold to address...	Auto	stage_order_detail	sold_to_address_id	General Nu...	Y

2. If the created columns are not relevant for the table, they can be deleted by using the right-click option to **Delete Column**.

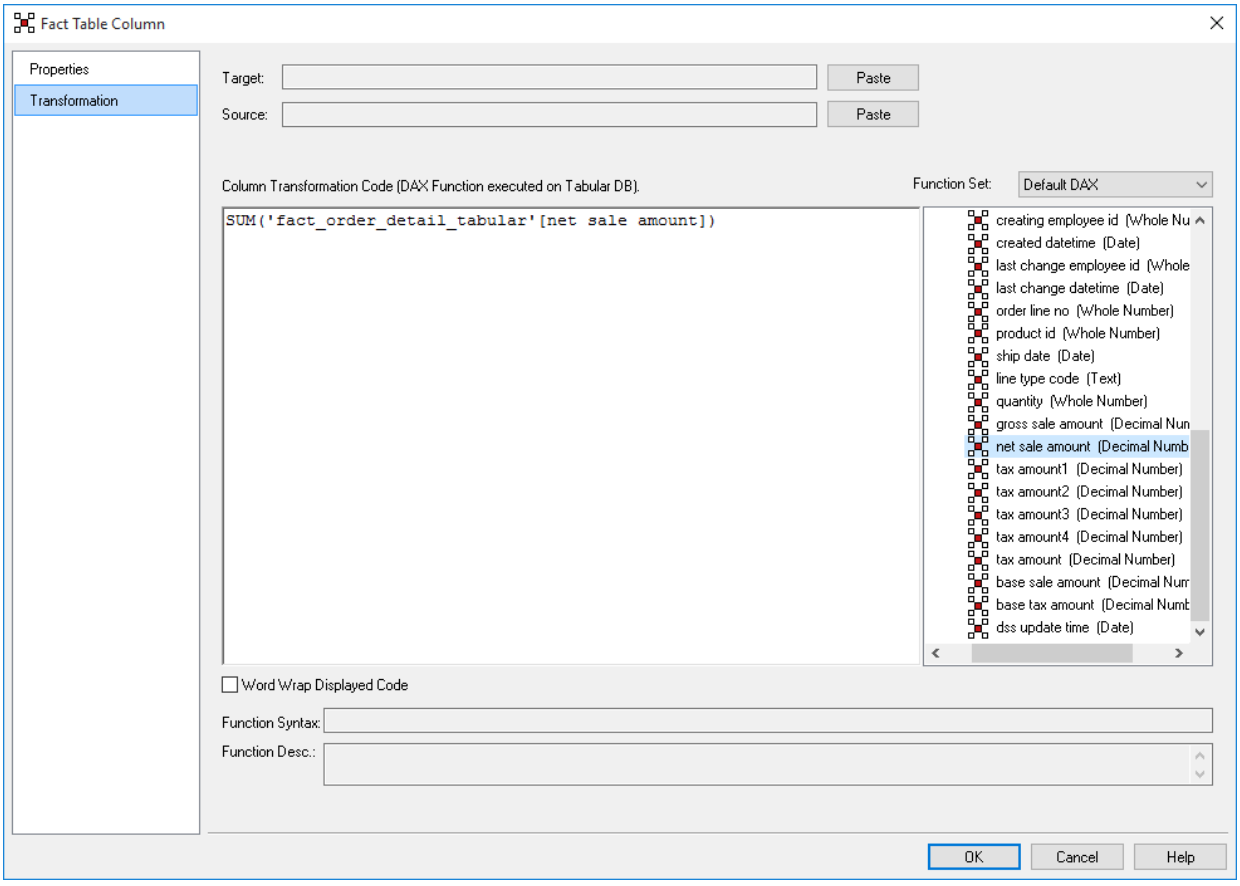


3. To add new **measures** to your Tabular Fact table, right-click on a column in the middle pane and click **Add Column**. In the table column properties you can set the following fields:
   - **Column name:** "measure_sales_total"
   - **Business Display Name:** "Sales Total"
   - **Type:** Measure

© Copyright, WhereScape Software Ltd All rights reserved.



4. Click the **Transformation** tab, add **SUM** to the Column Transformation Code and then select a sales column from the list. Click **OK**.



5. Right click the table **fact_order_detail_tabular** from the left pane and then select **Create(Recreate)**.

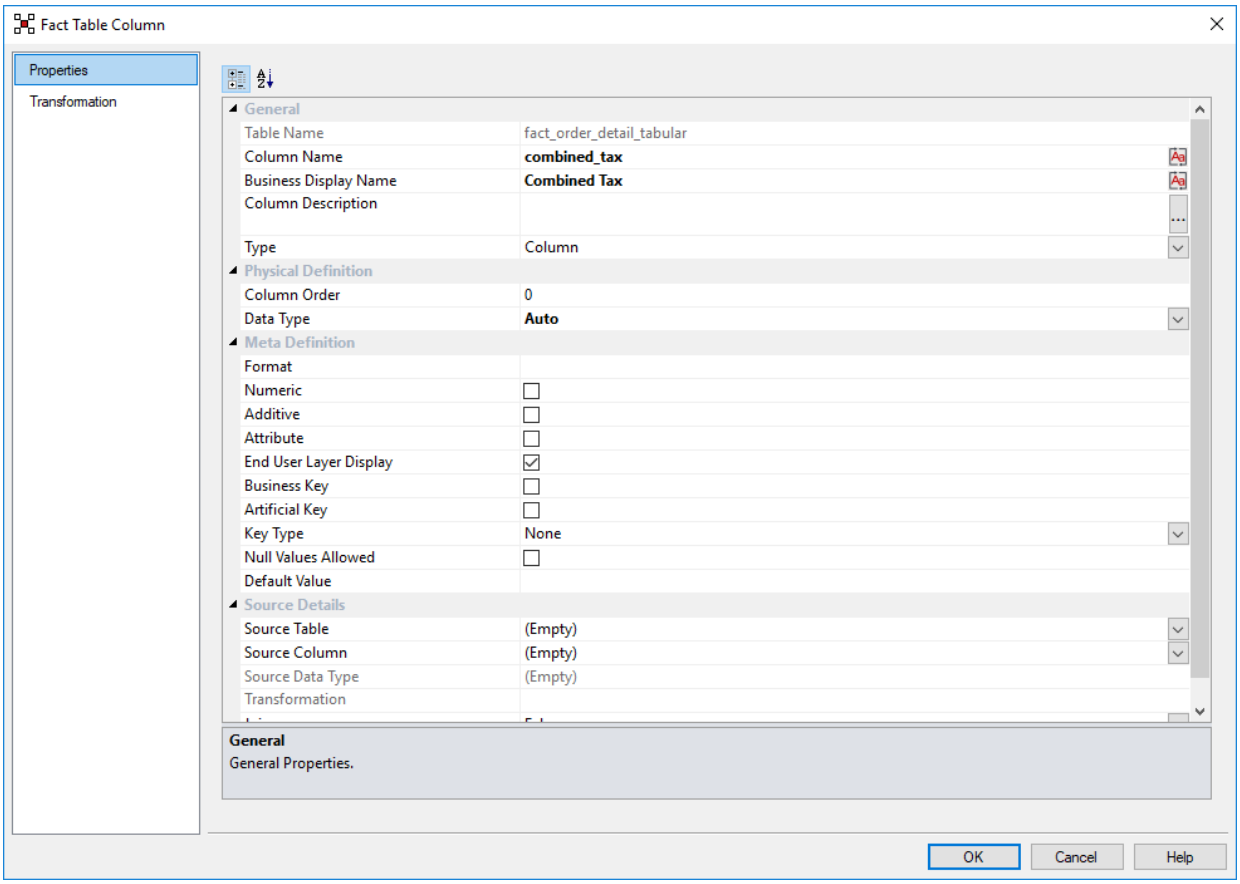
© Copyright, WhereScape Software Ltd All rights reserved.

6. After the table has been recreated, right click again and then select **Update** to update the table.

Tabular Model Calculated Columns

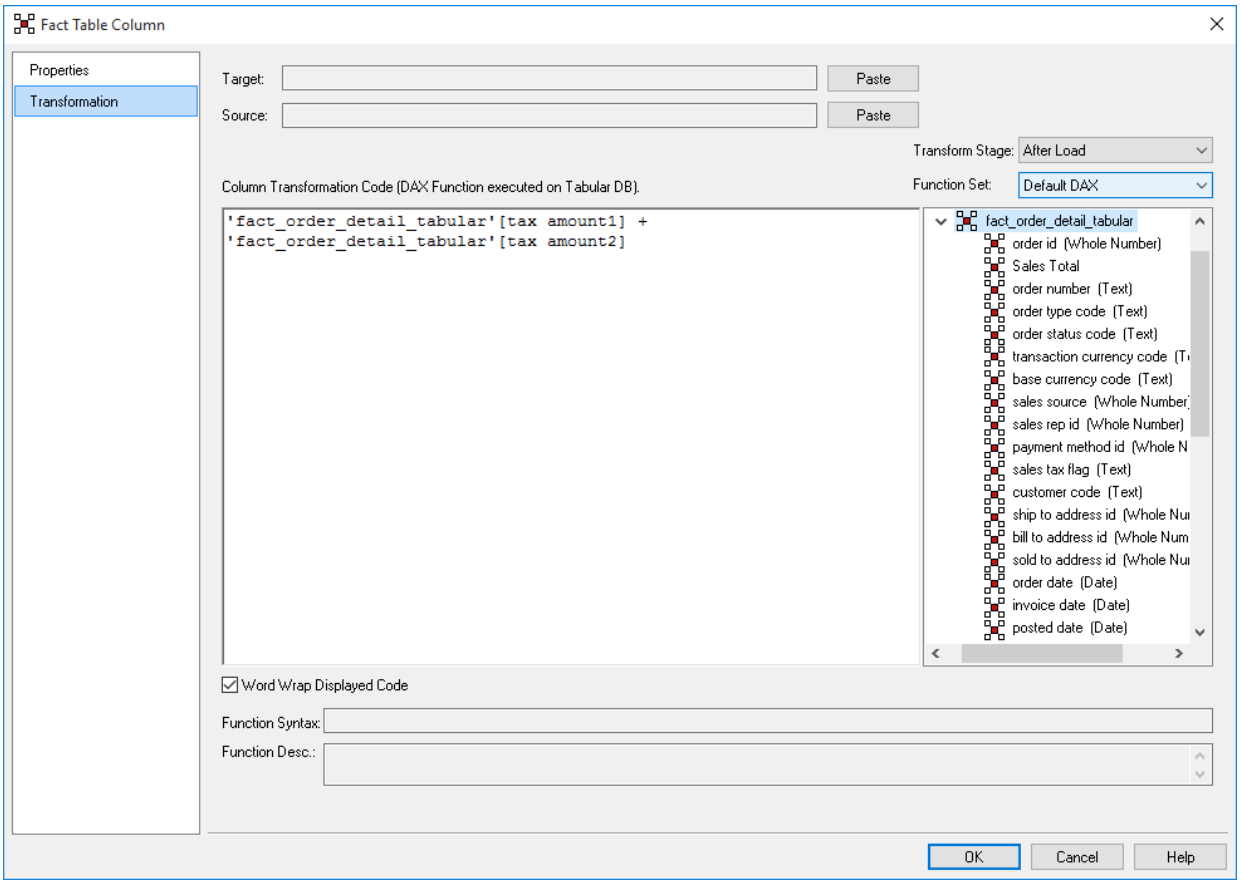
To add Calculated Columns to a Tabular Model:

1. Go to the Fact table's relevant column properties' screen to set the following fields:
   - **Column name:** "combined_tax"
   - **Business Display Name:** "Combined Tax"
   - **Type:** Column



2. Click the **Transformation** tab and select the following:
   - **After Load** as the Transformation Stage
   - **Default DAX** as the Function Set
   - Add a **SUM** and select the tax amount columns from the list of available columns on the right
   - Click **OK** on the fact column transformation screen

© Copyright, WhereScape Software Ltd All rights reserved.



Note:

Microsoft Analysis Services 2005+ Tabular Mode Tables: For Tabular Mode table column transformations, **Default DAX** is the only applicable Function Set for **after load** transformations.

3. Right click the table **fact_order_detail_tabular** from the left pane and then select **Create(Recreate)**.
4. After the table has been recreated, right click again and then select **Update** to update the table.

© Copyright, WhereScape Software Ltd All rights reserved.

Exporting Data

Export objects are used in WhereScape RED to produce ascii files from a single database table or view for a downstream feed. Some or all of the columns in a table or view can be exported.

There are three ways of performing exports in RED:

- **File export** - an export where most of the processing is managed and controlled by the scheduler
- **Script-based export** - an export where a Windows script file is executed to perform the export. Script-based exports on Windows supports both DOS Batch and **PowerShell scripts**.
- **Integration Services Export** - an export processed using a Windows connection where the processing is handled via an Integration Services Package that is generated and executed dynamically at run time. SSIS exports to UNIX/Linux connections and processed via the UNIX/Linux scheduler are currently not supported.

Building an Export Object

Tips

- The relevant client software for each database type must be installed on each machine you want to export to.
- For **Integration Services** exports, ensure the **SSIS Version** is selected in **Tools > Options > Code Generation > SSIS Version.**

Note

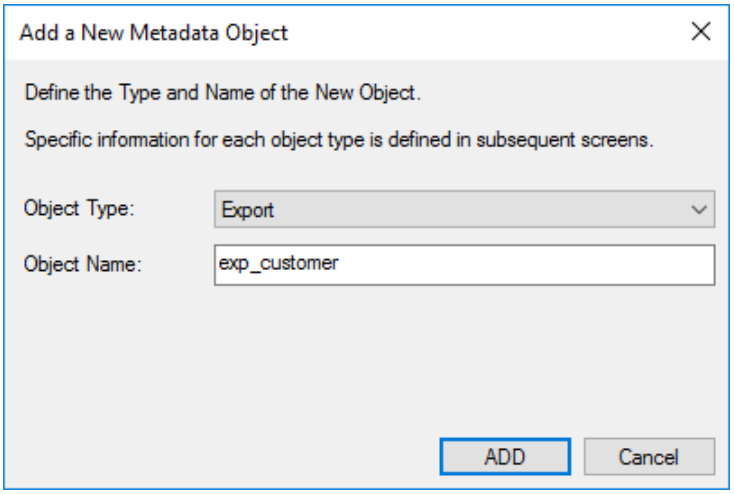
Greenplum

File Exports are the only export type currently supported in Greenplum from both Windows and UNIX/Linux connections. Script-based exports are currently unsupported.

The simplest way to create an export object is to use the **drag and drop** functionality of WhereScape RED.

Creating an Export:

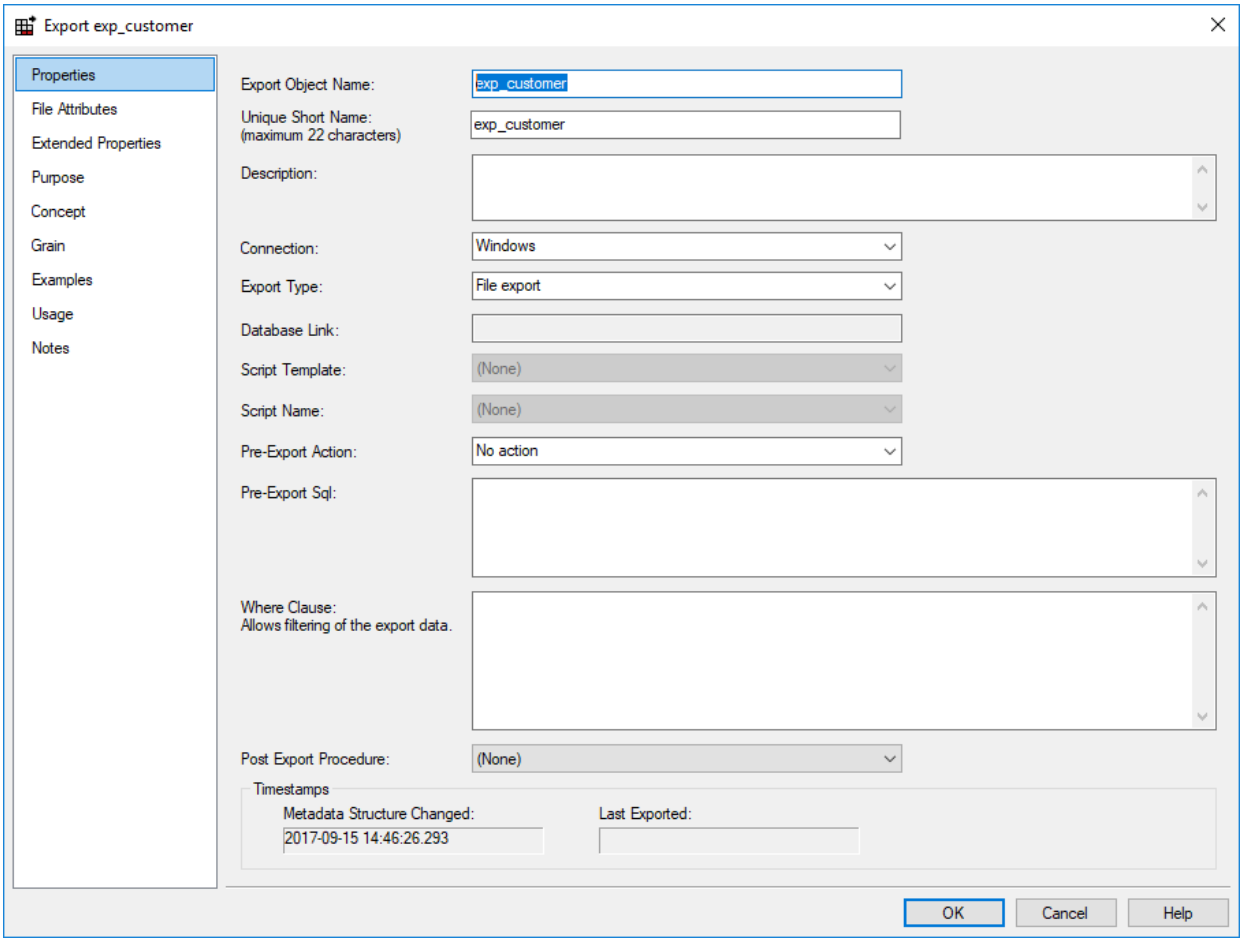
1. Browse to the Datawarehouse connection (**Browse > Source Tables**).
2. Create a drop target by double-clicking on the **Export** object group in the left pane. The middle pane should have a column heading of **Export Objects** for the leftmost column.
3. Select a table or view in the right pane and drag it into the middle pane. Drop the table or view anywhere in the middle pane. The following dialog appears:



4. If the export object needs to be renamed, rename it and then click **ADD**.

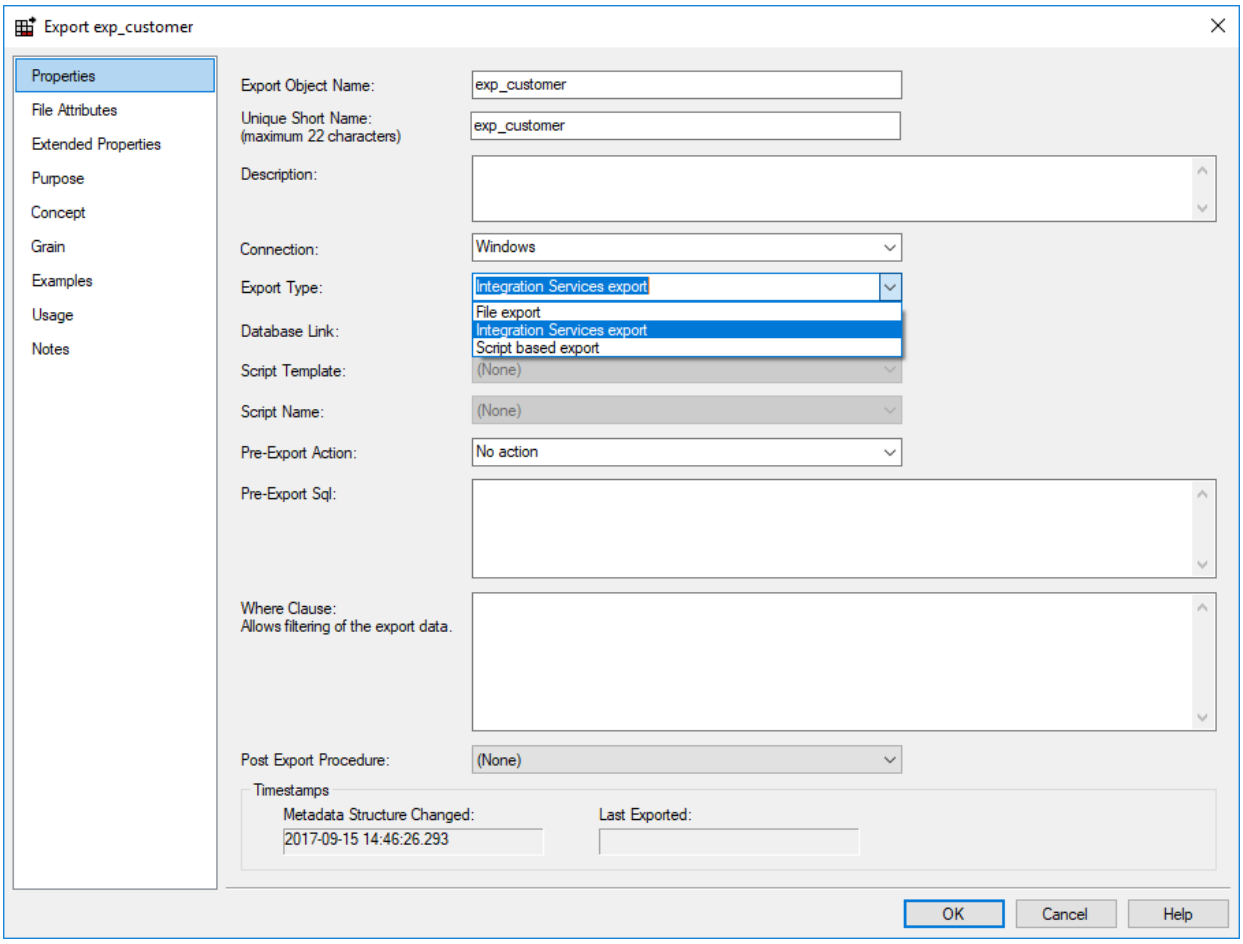
© Copyright, WhereScape Software Ltd All rights reserved.

The Properties window appears.



5. Select the **Connection** that you want to perform the export from the Connection drop-down list. In this example, the Connection is **Windows**.
    - Select the preferred Export type: **File export**, **Script-based export**
    - **Integration Services Export** is also an available export type when exporting objects using a **Windows** connection.

© Copyright, WhereScape Software Ltd All rights reserved.



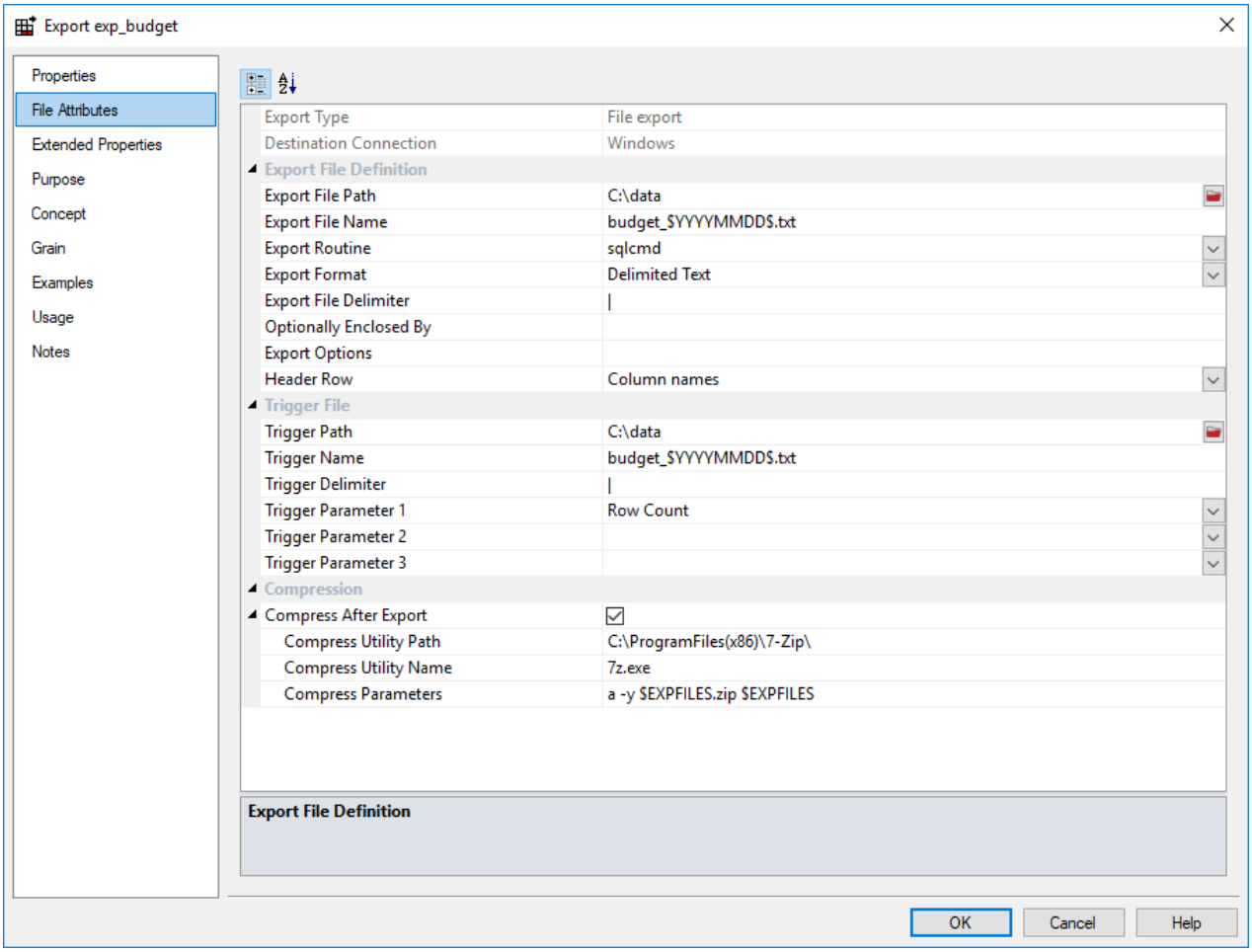
6. Click the **File Attributes** tab and fill in the fields as described in the next section.
7. Finally, run the export by right-clicking it and selecting **Export**.

© Copyright, WhereScape Software Ltd All rights reserved.

File Attributes

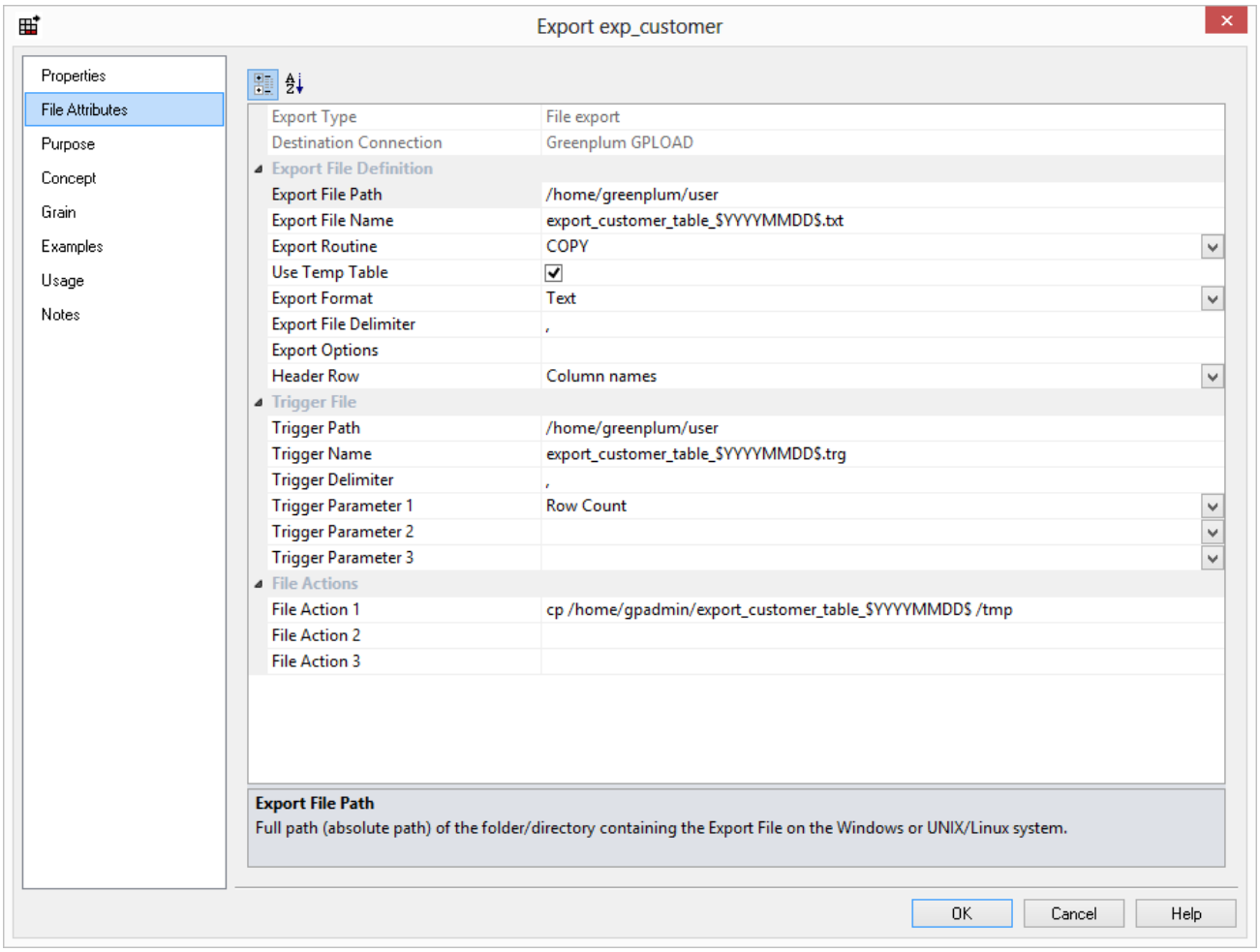
The following fields are available to define the location, name and contents of the exported data file:

SQL Server File Attributes example screen:



© Copyright, WhereScape Software Ltd All rights reserved.

Greenplum File Attributes example screen:

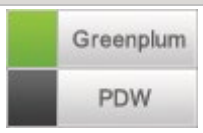
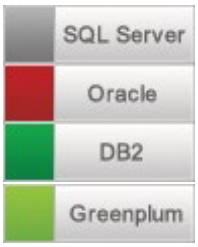


Fields	Description
Export Type	Method of exporting data from the table. The available options are dependent on the Destination connection that can be specified via the Properties page.
Destination Connection	Destination to the file system to which data will be exported. The destination connection can be specified on the Properties screen.

Export File Definition

Fields	Description
Export File Path	The full path (absolute path) of the folder/directory where the File is to be created on the Windows or UNIX/Linux system.
Export File Name	Name of the Export File to which the data will be exported. The variable $SEQUENCE$ can be used to provide a unique sequence number for the export file. Also the data/file components YYYY, MM, HH, MI, SS can be used when enclosed with the $ character.For example an export file name might be budget_$YYYYMMDD$.txt which would result in a file name like budget_20150520.txt.
Export Routine	Database-specific routine to use to export the data. SQL Server In SQL Server this is BCP or isql. In later versions of SQL Server this is BCP or sqlcmd. Oracle For Oracle this is SQL*Plus. DB2 For DB2 the supported export routine is EXPORT.

© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
	 For Greenplum, the supported export routine is COPY. For PDW, the supported export routine is sqlcmd.
Use Temp Table	This option is only available for **Greenplum** and is enabled by default. Ticking this check-box creates a temporary table to export the data into. The temporary table is then deleted once the data has been exported.
Export Format	Routine-specific format to use to export the data.  This is the format of the data in the export file and is either Delimited Text or Width Fixed Text in Oracle and SQL Server. In DB2, the supported formats are DEL (delimited), WSF (worksheet format) and IXF(intergrated exchange format). In Greenplum, the supported formats are Text and CSV.
Export File Delimiter	Character that separates the fields within each record of the Export File for Delimited formats. The delimiter identifies the end of which field. Common field delimiters are tab, comma, colon, semi-colon, pipe. To enter a special character enter the uppercase string CHAR with the ASCII value in brackets (e.g. CHAR(9)). This is only available if the Export Format is Delimited Text.
Optionally Enclosed by	Character that brackets text fields within each record of the Export File for Delimited formats. A common example is ". This is only available if the Export Format is Delimited Text.
Header Row in Export	If a header line is required, choose business names or column names from this drop-down list. This option is not available in DB2.
Export Options	Allows the entry of export utility options. If more than one option is required, then a semi-colon should be used between options. This option is not available in DB2.
Export Modifier	DB2 only. This allows the entry of EXPORT modifiers in DB2.

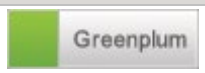
Trigger File

Fields	Description
Trigger file	If a trigger file is specified then this file is created after the export file. It will normally contain a row count and a check sum.
Trigger Path	The purpose of the trigger file is to indicate that the export to the main file has completed and that it is now safe to load the file. Secondly the trigger file may contain control sums to validate the contents of the main load file. This field should contain the full path name to the directory in which a trigger file is to be generated on the destination system.
Trigger Name	Refers to the name of the file that is to be created as a trigger file. A trigger file typically contains check sums (row count or the sum of a numeric column). The variable $SEQUENCE$ can be used to provide a unique sequence number for the trigger file. Also the data/file components YYYY, MM, HH, MI, SS can be used when enclosed with the $ character. For example a trigger file name might be budget_$YYYYMMDD$.txt which would result in a file name like budget_20150520.txt.
Trigger Delimiter	Multiple fields in the trigger file are to be separated by the trigger delimiter.
Trigger Parameter 1,2,3	The checksums to be put in the trigger file. One of the row count and the sum of any numeric fields in the source data.

© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description	
	Compress After Export	Check this box if you want to compress the export file after it has been created.
	Compress Utility Path	The directory in which the compress utility exists.
	Compress Utility Name	The name of the compression utility executable.
	Compress Parameters	The name of the file to be compressed (using the WhereScape RED variable $EXPFILE$) and any commands or switches required to make the compression utility work. These parameters will depend on the compression utility used.

File Actions

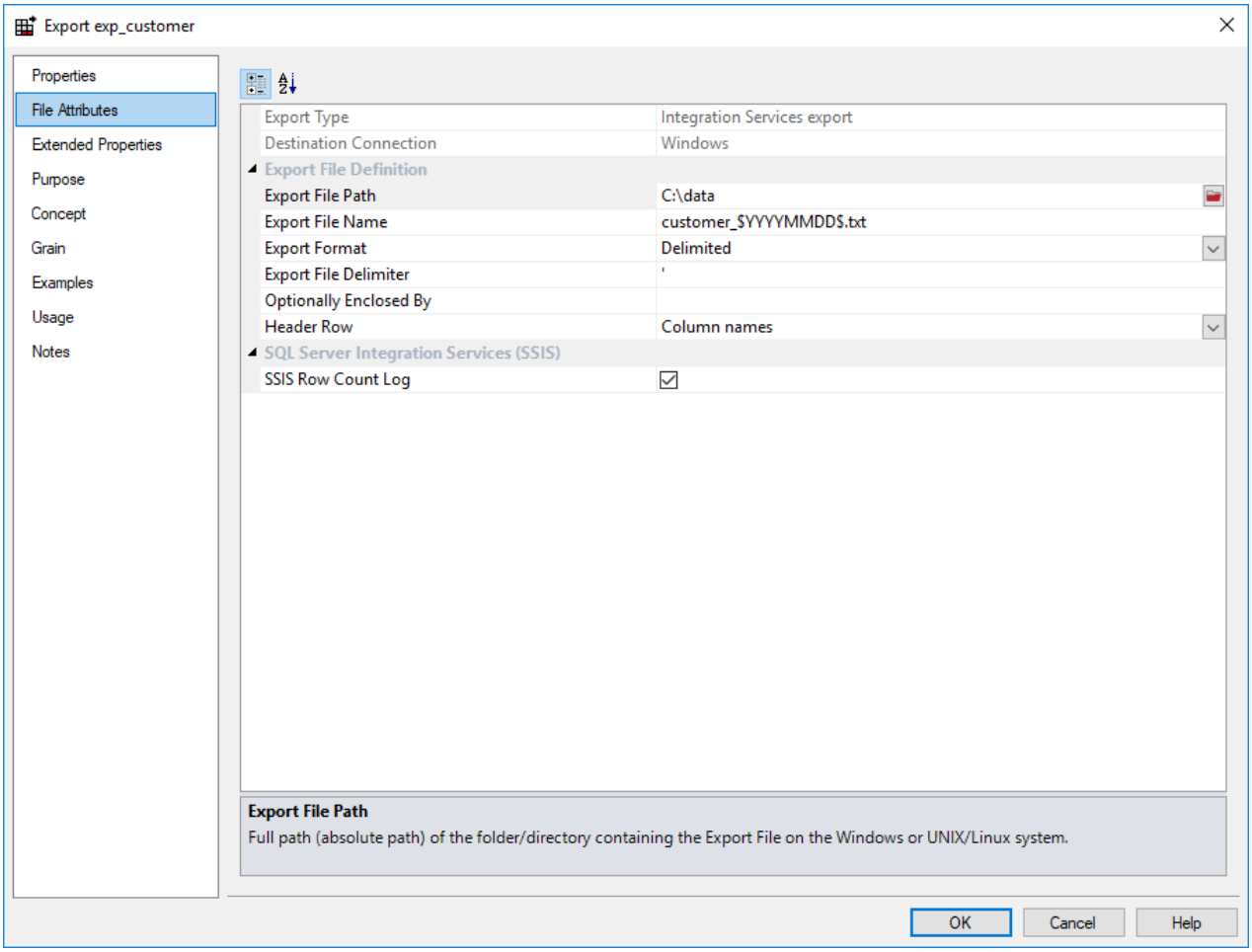
Fields	Description	
File Action 1, 2, 3		File Actions are only available for Greenplum. Use these fields to enter any command lines to run after the export such as copying files to another machine or deleting files.

© Copyright, WhereScape Software Ltd All rights reserved.

File Attributes - SSIS Exports

The following fields below are available to define the location, name and definitions of the exported data file:

SQL Server File Attributes example screen



Fields	Description
Export Type	Method of exporting data from the table. The available options are dependent on the Destination connection that can be specified via the Properties page.
Destination Connection	Destination to the file system to which data will be exported. The destination connection can be specified on the Properties screen.

Export File Definition

Fields	Description
Export File Path	The full path (absolute path) of the folder/directory/ where the File is to be created on the Windows or UNIX/Linux system.
Export File Name	Name of the Export File to which the data will be exported. The variable $SEQUENCE$ can be used to provide a unique sequence number for the export file. Also the data/file components YYYY, MM, HH, MI, SS can be used when enclosed with the $ character. For example an export file name might be customer_$YYYYMMDD$.txt which would result in a file name like customer_20150520.txt.
Export Format	Routine-specific format to use to export the data.
Export File Delimiter	Character that separates the fields within each record of the Export File for Delimited formats. The delimiter identifies the end of which field. Common field delimiters are tab, comma, colon, semi-colon, pipe. To enter a special character enter the uppercase string CHAR with the ASCII value in brackets (e.g. CHAR(9)). This is only available if the

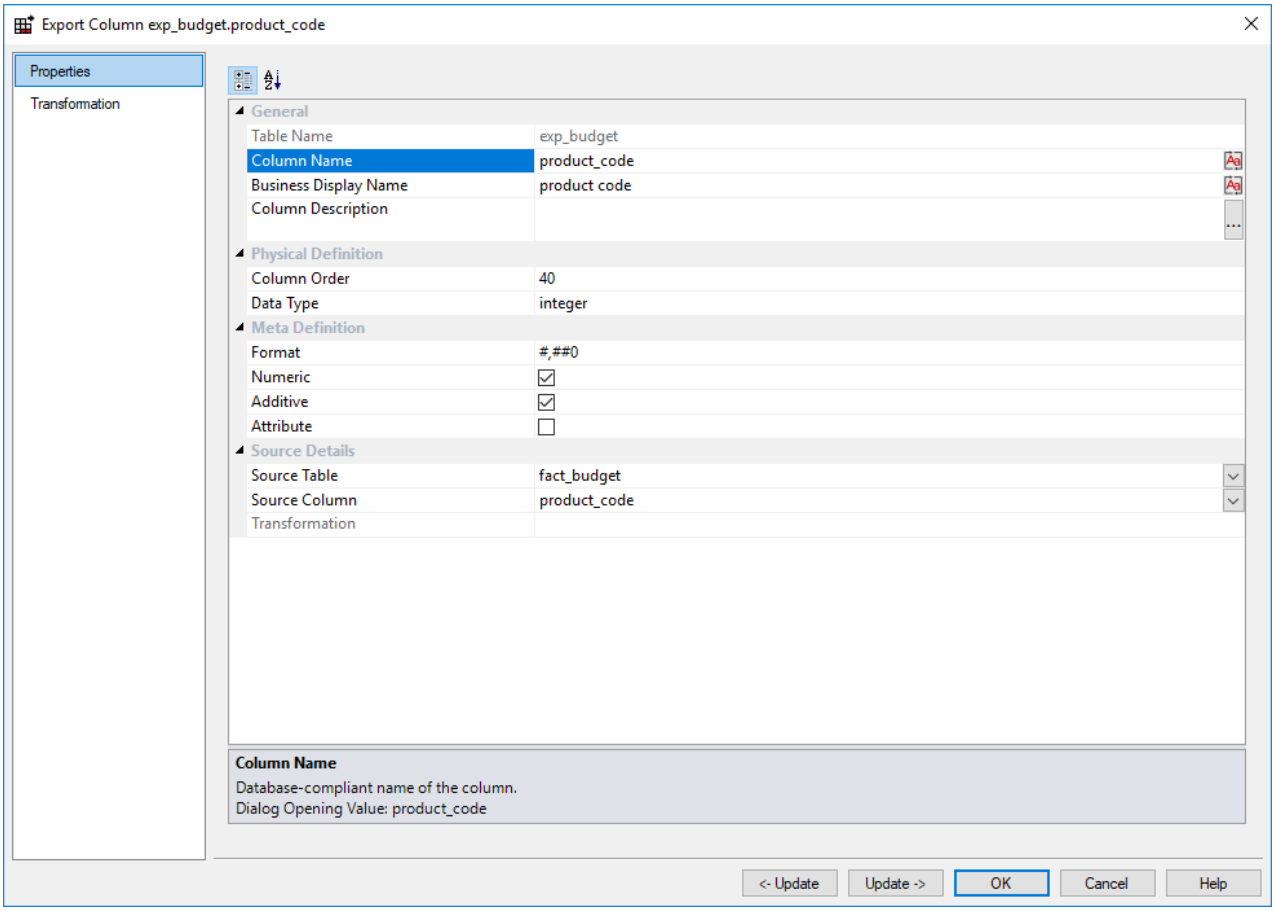
© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
	Export Format is Delimited Text.
Optionally Enclosed by	Character that brackets text fields within each record of the Export File for Delimited formats. A common example is ". This is only available if the Export Format is Delimited Text.
Header Row	If a header line is required, choose business names or column names from this drop-down list. This option is not available in DB2.

SQL Server Integration Services (SSIS)

Fields	Description
SSIS Row Count Log	When enabled, this option includes Row Count logging on SSIS exports.

Export Column Properties



General

Fields	Description
Table Name	Database-compliant name of the table that contains the column. [Read-only].
Column Name	Database-compliant name of the column. Typically column-naming standards exclude the use of spaces etc. A good practice is to only use alphanumerics, and the underscore character. Changing this field alters the table's definition. **Note** A case conversion button on the right converts the text between different cases: UPPER CASE, Capitalized Case and lower case. The mode cycles to the next case in

© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
	the sequence each conversion.
Business Display Name	Name that the business uses to refer to the column, which is included in the RED-generated documentation and can be used in the end user layer of other tools. [Does NOT affect the physical database table]. As such it is a free form entry and any characters are valid. **Note** A case conversion button on the right converts the text between different cases: UPPER CASE, Capitalized Case and lower case. The mode cycles to the next case in the sequence each conversion.
Column Description	This field contains the description for the column. It may be a description from a business user's point of view. This field might additionally contain information on where and how the column was acquired. For example if the column is sourced from multiple tables or is a composite or derived column then this definition would normally describe the process used to populate the column. This field is used in the documentation and is available via the view **ws_admin_v_dim_col .** This field is also stored as a comment against the column in the database.

Physical Definition

Fields	Description
Column Order	Numeric value that controls the relative order of columns in the database create statement. The lowest numbered column will appear first in the table. Although this affects the physical table definition no action will be taken unless the table is re-created in the database. The columns can be re-numbered based on the existing order by choosing the **Respace Order Number** pop-up menu option when positioned over any column in the table. This action will number the columns in increments of 10 starting at 10. In addition to a simple change of the order field, the column order can be changed by first displaying the columns in the middle pane and then using drag and drop to move the columns around. This drag and drop process will automatically renumber the columns as required.
Data Type	Database-compliant data type that must be valid for the target database. Typical data types for Oracle are integer, number, char(), varchar2() and date. For SQL Server common types are integer, numeric, varchar() and datetime. Refer to the database documentation for a description of the data types available. Changing this field alters the table's definition.

Meta Definition

Fields	Description
Format	Not relevant for Export Objects.
Numeric	Not relevant for Export Objects.
Additive	Not relevant for Export Objects.
Attribute	Not relevant for Export Objects.

Source Details

Fields	Description
Source Table	Identifies the source table where the column's data comes from. This source table is normally a stage table within the data warehouse. If the column was sourced from multiple tables, then the normal practice is to record one of the tables in this field and a comment listing all of the other tables in the **Source Strategy** field. This field is used when generating a procedure to update the fact table. It is also used in the track back

© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
	diagrams and in the documentation.
Source Column	Identifies the source column where the column's data comes from. Such a column is normally a stage table column, which in turn may have been a transformation or the combination of multiple columns. This may also be a dimensional key where a dimension is being joined.
Transformation	Transformation. [Read-only].

© Copyright, WhereScape Software Ltd All rights reserved.

Script Based Exports

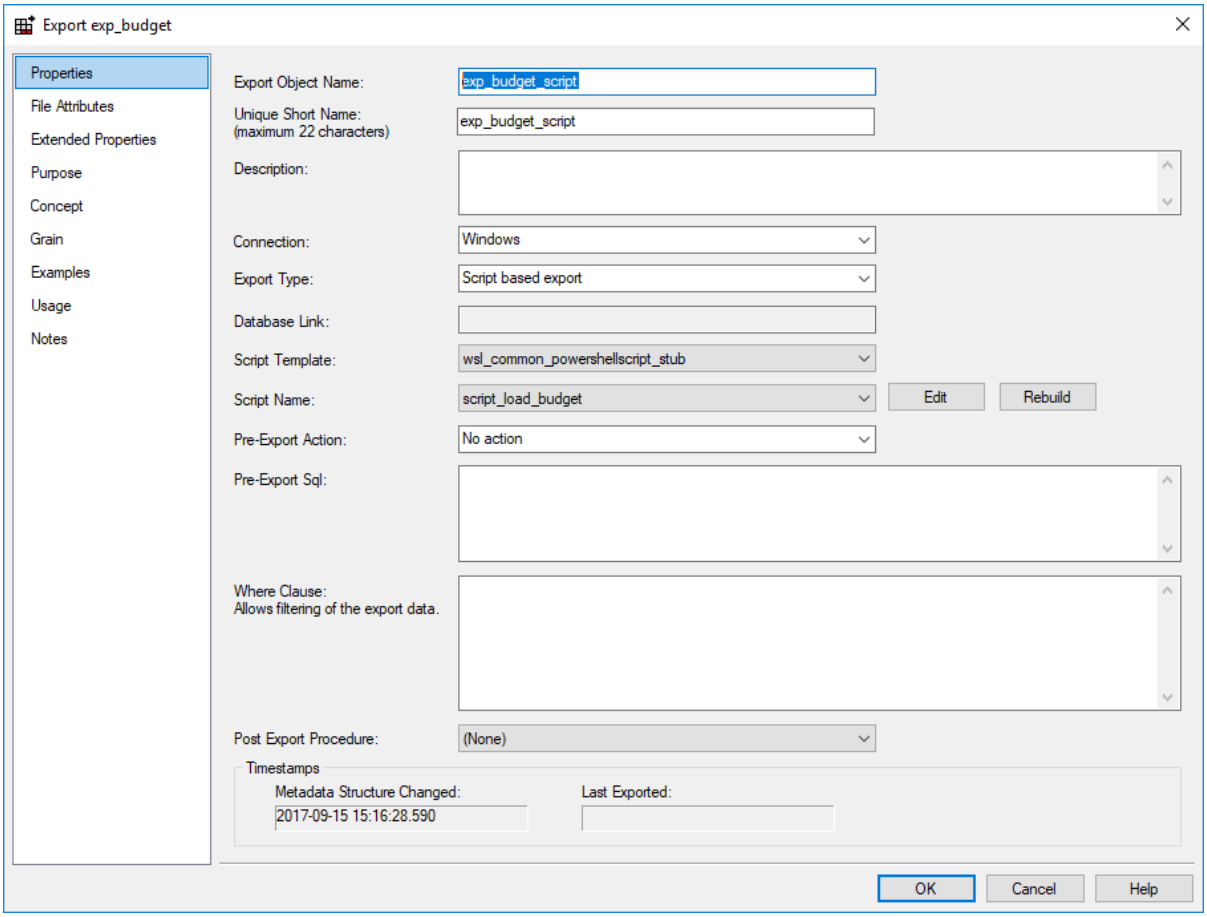
A script based export object will have a Host Script defined. During the export process, the host script is executed and the results returned.
During the **drag and drop** creation of an export object from a single table or view, a script can be generated by selecting one of the 'Script based' export options. This script can then be edited to more fully meet any requirements.

Note

Script-based exports on Windows supports both DOS Batch and **PowerShell scripts**.

There are a number of conventions that must be followed if these host scripts are to be used by the RED Scheduler.

1. The first line of data in **standard out** must contain the resultant status of the script. Valid values are '1' to indicate success, '-1' to indicate a Warning condition occurred but the result is considered a success, '-2' to indicate a handled Error occurred and subsequent dependent tasks should be held, -3 to indicate an unhandled Failure and that subsequent dependent tasks should be held.
2. The second line of data in **standard out** must contain a resultant message of no more than 256 characters.
3. Any subsequent lines in **standard out** are considered informational and are recorded in the audit trail. The normal practice is to place a minimum of information in the audit trail. All bulk information should be output to **standard error.**
4. Any data output to **standard error** is written in the error/detail log. Both the audit log and detail log can be viewed from the WhereScape RED tool under the Scheduler window.
5. When performing **Script based exports**, it is easy to use the **Rebuild** button beside Script Name field to rebuild the scripts.



© Copyright, WhereScape Software Ltd All rights reserved.

Transformations

Standard **column transformations** can be used in WhereScape RED to perform calculations, change data types or format data.

Re-using complex transformations can save a significant amount of time. These can be achieved in the following ways with WhereScape RED:

- Database Functions
- WhereScape RED Re-usable Transformations
- Teradata User Defined Functions (UDFs)

Column Transformations

Each table or export object column can have a transformation associated with it. For all table types, except for load tables, the transformation will be included in the generated procedure for the table. These are executed as part of the procedure update. The transformation must therefore be a valid SQL construct that can be included in a **Select** statement.
For example, we could have a transformation of 'load_order_line.quantity * 0.125' to calculate a tax column of 12.5%. Click the **Transformation** tab on the Column Properties window to enter a transformation.

Notes:

- Transformations added to an existing table object that have an update procedure are only put into effect when the procedure is re-generated and re-compiled.
- A common transformation can be applied to several columns via the **Change Columns** window.

Column transformations in Load tables are more complex, due to the unique nature of Load tables. Refer to **Load Table Column Transformations** for details.

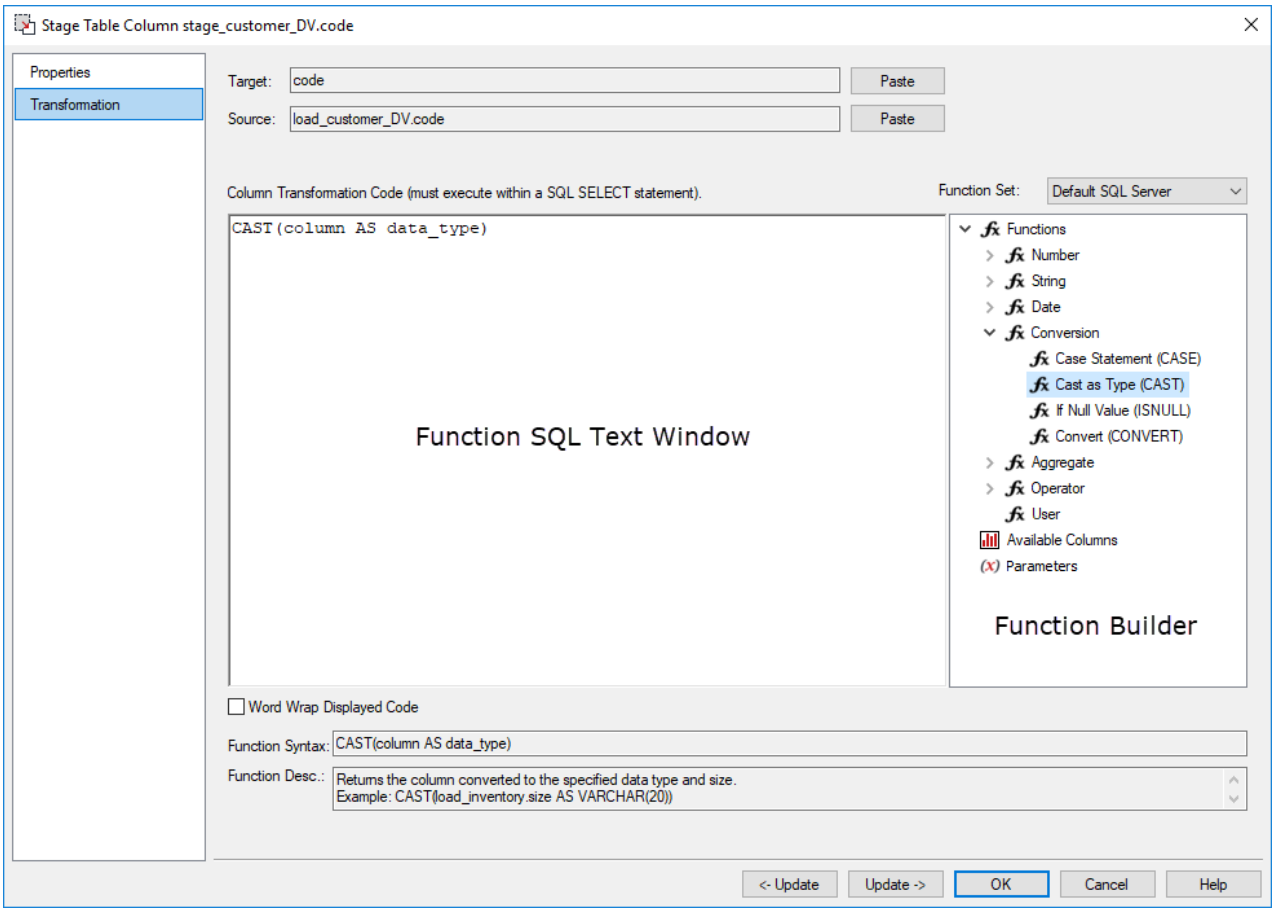
Dimension View transformations are included in the Database Definition Language (DDL) that creates the view in the database. Any changes to Dimension View column transformations require the view to be dropped and recreated.

Export object column transformations are dynamically applied for file exports. If the export object is executed via a Host Script, then the script needs to be regenerated for changes to transformations to take effect. Refer to **Exporting Data** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

Column Transformation Properties

An example below shows the transformation **Properties** screen with a simple transformation:



The two special update keys allow you to update the column and step either forward or backward to the next column's properties. **ALT-Left Arrow** and **ALT-Right Arrow** can also be used instead of the two special update keys.

Fields	Description
Function SQL Text Window	The **Function SQL Text Window** contains the SQL used in the transformation. It can be directly entered, built up using the **Function Builder** and **Add** buttons or a combination of both.
Function Builder	The **Function Builder** contains a list of standard database functions, operators, user defined functions, all columns belonging to all source tables and **parameters** defined in the meta data repository. • Expanding the **Function Heading** displays the **Function Groups** (Number, String, Data, Conversion, etc.) and the **Re-usable Function Heading**. Similarly, expanding the other headings displays the **Available Source Columns** or **Parameters**. • Each function group, source column or parameter can in turn be expanded to show individual **Functions** , **Source Columns** and **Parameters**. • Double-clicking on a function adds the **Function Model** to the **Function SQL Text Window**. The first variable (almost always the source column) is left highlighted in the **Function SQL Text Window**. You can insert additional Functions, Source Columns or Parameters in the correct place within the **Function SQL Text Window** by double-clicking them from the **Function Builder**pane.
Target Paste Button	The **Target Paste** button adds the current column in the form ColumnName to the **Function SQL Text Window** at the location of the cursor.

© Copyright, WhereScape Software Ltd All rights reserved.

Fields	Description
Source Paste Button	The **Source Paste** button adds the source table and column in the form TableName.ColumnName to the **Function SQL Text Window** at the location of the cursor.
Transform Stage	Only visible on Load table column transformations. Refer to **Load Table Column Transformations** for details.
Function Set	This drop-down list enables the user to select which set of functions are to be displayed in the tree view when creating a transformation on a column of a table.
Update Buttons	The Update Buttons: **Update <-** and **Update ->** are used to move from the current column to previous and next columns respectively in the current table. The alternative is to exit the Column Transformation Properties, choose the next column, re-enter the Column Properties and choose the **Transformation** tab.
Function Syntax	The syntax guide for the **Function,** visible when you click on the function. Essentially the same as the Function model; loaded into the **Function SQL Text Window** when you click on the function. Read only.
Function Desc	The description of the **Function** visible when the function is clicked. Read only.
Function Model	The model (template SQL code) for a **Re-usable Transformations**. This is only visible for Re-usable Transformations. Read only.
Localize Transformation	The **Localize transformation** button breaks the link between a **column transformation** and the **Re-usable Transformation** it is based on. If this button is clicked, changes to the underlying re-usable transformation cannot be automatically propagated to the column transformation. This is only visible for Re-usable Transformations.

Load Table Column Transformations

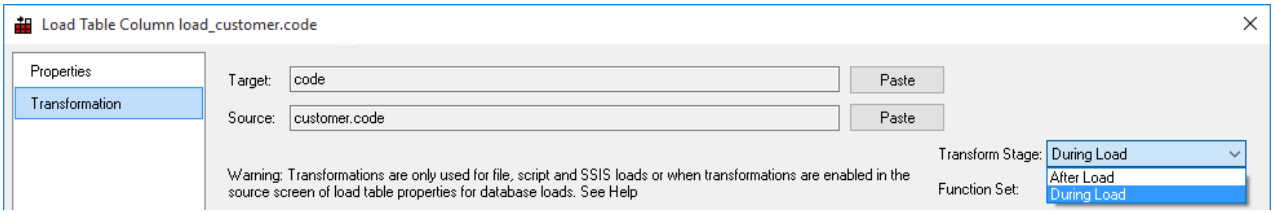
Data entering the data warehouse can be manipulated if required. This manipulation can occur at any stage, but is supported via a number of methods during the **Load** stage. Load tables provide options to transform data. If multiple pass transformations are required, then a Load table can be created from another Load table, e.g. multiple Load tables can be supported in the data flow path.

The options available differ depending on the type of load but in most cases the **after** transformation and post load procedure can be utilized. Specifically:

Database Link Load	• During Load transformations • After Load transformations • Post Load procedure
ODBC Based Load	• During Load transformations • After Load transformations • Post load procedure
File Based Load	• After Load transformations • Post load procedure
Integration Services Load	• During Load transformations • After Load transformations
Script Based Load	• During Load transformations • After Load transformations • Post load procedure
Externally Loaded	• After Load transformations • Post Load procedure

© Copyright, WhereScape Software Ltd All rights reserved.

The **Transformation** tab of a column's properties is used to define **during** and **after** load transformations. It can only be one or the other for a specific column. One column can be used to build another, so an **after** can be based on the results of a **during**, if different columns are used.



Note

The **During** transformations use **Source Table** columns. The **After** transformations use the **Load Table** columns.

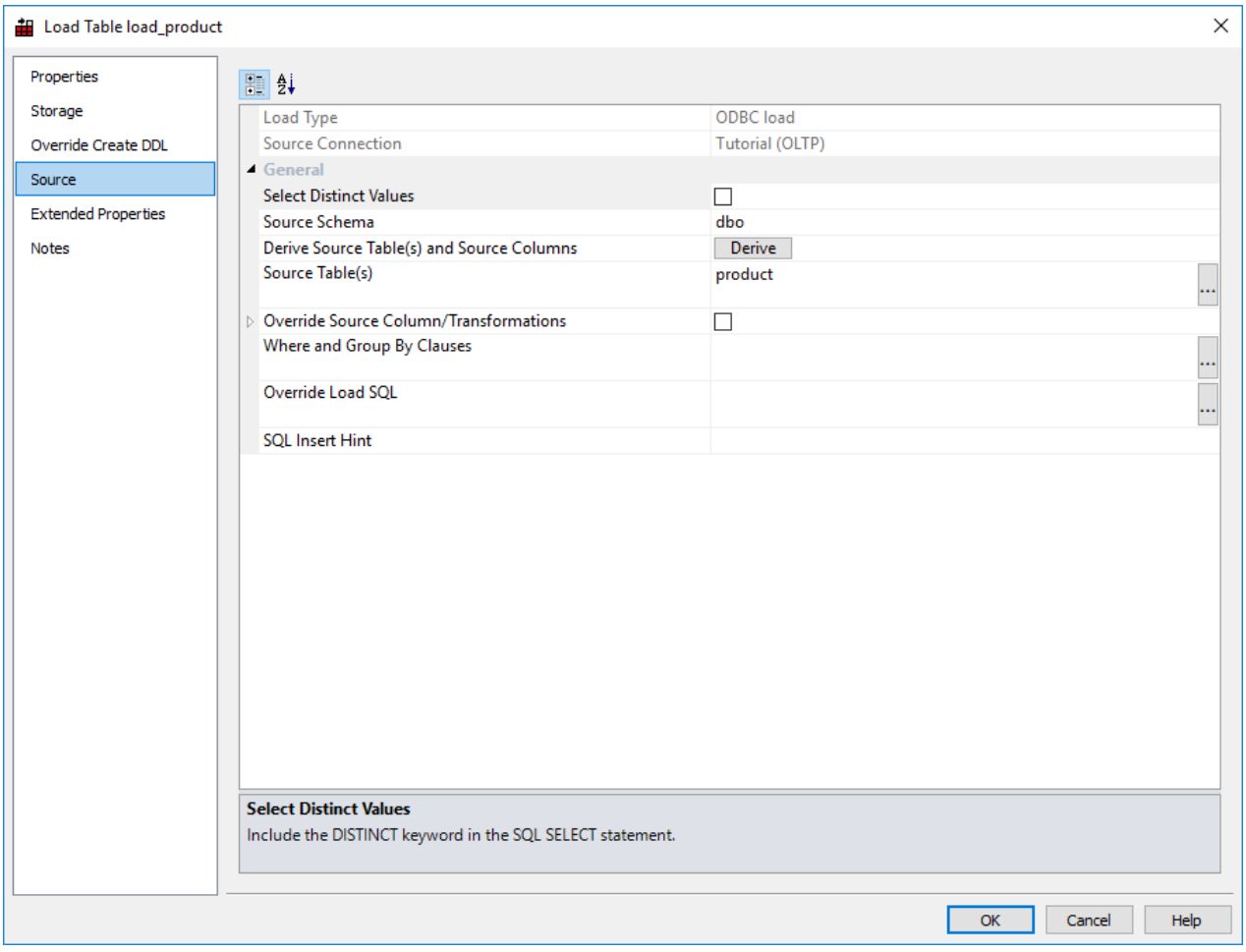
All **After** transformations take place in the native SQL language of the data warehouse database. The **During** transformations differ in terms of which language is used. This is particularly true for file based loads. Normally the 'During' transformation will occur in the native SQL language of the source database.

For **Flat file loads** using SSIS, **After Load transformations** use the SQL syntax of the Target database but **During Load transformations** use SSIS expression syntax that can be referred to on the Microsoft Developer Network website: https://docs.microsoft.com/en-us/sql/integration-services/expressions/syntax-ssis?view=sql-server-2017.

Database Link During Load Transformations

The **during** load transformation allows the manipulation of column data as it enters the data warehouse.

By default, **Database link loads** and **ODBC based loads** have **During** and **After** transformations enabled.



© Copyright, WhereScape Software Ltd All rights reserved.

When transformations are enabled, the contents of a source table/source column for each column are used as the basis of the loading statement.

- If source table and source column are null, then a null is used.
- If data exists in the **Transformation** tab of a column's properties, then this transformation data is used instead of the source table/source column combination.

Example

The following load table columns will generate the load sql statement if no transformation data is present against these columns.

load_product Columns

Column Name	Display Name	Data Type	Source Table	Source Column
code	code	numeric(6)	product	code
description	description	varchar(64)	product	description
prod_line	prod line	varchar(24)	product	prod_line
prod_group	prod group	varchar(24)	product	prod_group
subgroup	subgroup	varchar(24)	product	subgroup

The SQL code from the load results is:

Results

Object	Message
load_product	Load Complete
load_product	Table dbo.load_product truncated.
Load	INSERT INTO dbo.load_product WITH (TABLOCK)
Load	SELECT
Load	product.code,product.description,product.prod_line,product.prod_group,product....
Load	From [Tutorial].dbo.product product
Load	;
Load	9 rows loaded

Results Reports

If the Column has a transformation defined as follows:

UPPER(substring(description,1,1)) + LOWER(substring(description,2,1))

then the following SQL statement will be executed.

Results

Object	Message
load_product	Table dbo.load_product truncated.
Load	INSERT INTO dbo.load_product WITH (TABLOCK)
Load	SELECT
Load	product.code,UPPER(substring(description,1,1)) + LOWER(substring(description,
Load	2,1)),product.prod_line,product.prod_group,product.subgroup
Load	From [Tutorial].dbo.product product
Load	;
Load	9 rows loaded

Results Reports

© Copyright, WhereScape Software Ltd All rights reserved.

File During Load Transformations

The loading of flat files is performed using the database specific loader. For Oracle, this is SQL*LOADER. For SQL Server it is the Bulk Insert Transact SQL statement. The contents of the Transformation tab in the column's Properties are the functions and conversions supported by the database loader.

Example

SQL*LOADER performs transformations such as:

- DATE 'DD-MMM-YYYY'converts from a value such as 23-Mar-1999 to an Oracle date.
- "TO_NUMBER(:COL)"converts the data to a number
- "NVL(TRIM(:COL),0)"trims leading and trailing white characters and inserts zero if null.

In Teradata, the loading of flat files is performed using Fastload, Multi-load or TPT. The contents of the Transformation tab in a column's Properties are the functions and conversions supported by the database loader.

Example

- Multiload performs transformations such as:
- FORMAT 'DD-MMM-YYYY' converts from a value such as 23-Mar-1999 to a Teradata date.
- CAST(:COL AS NUMERIC(18,4)) converts the data to a numeric(18,4)
- COALESCE(LTRIM(RTRIM(:COL)),0) trims leading and trailing white characters and inserts zero if null.

After Load Transformations

After transformations will initiate a pass of the Load table after the load has occurred. They enable manipulation of the Load table columns using the database and SQL functions.

SQL Server Example:

The following **after** transformation set for the column code in the table load_product

lpad(code,5,'0')

would result in the following SQL statement being executed after the load:

update load_product set code = lpad(code,5,'0');

Teradata Example:

The following **after** transformation set for the column code in the table load_product

substrlpad(code,1,5)

would result in the following SQL statement being executed after the load:

update load_product set code = substrcode,1,5);

© Copyright, WhereScape Software Ltd All rights reserved.

Database Functions

Database functions can be created in the data warehouse database (as procedure objects in WhereScape RED) and used in column transformation in WhereScape RED. Refer to the Oracle and SQL Server documentation on functions and **Procedures and Scripts** for more information on creating functions inside WhereScape RED.

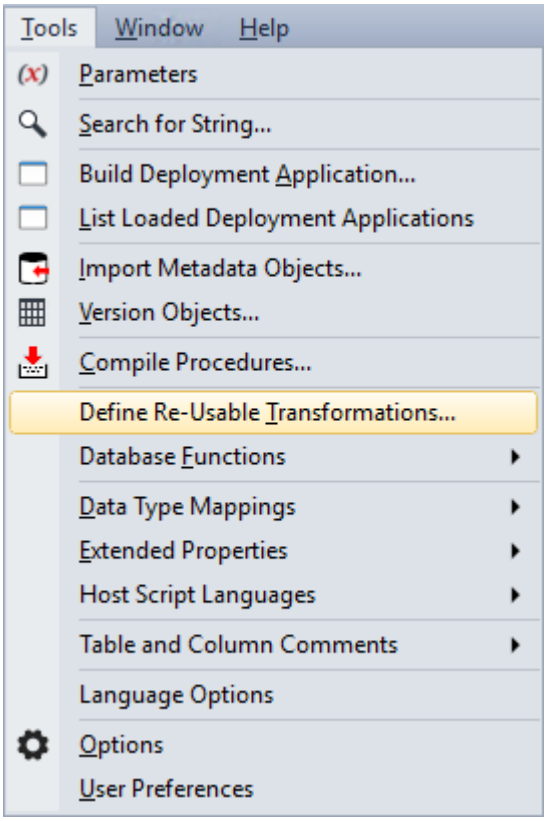
Database function sets can also be created, edited, deleted, imported and exported, using the RED **Database Function sets** tool.

Re-usable Transformations

WhereScape RED Re-usable Transformations allow a complex SQL transformation using standard database functions (including User Defined Functions) to be built once and reused in multiple column transformations. Functionality is included to manage and propagate changes to re-usable transformations.

Creating a New Re-usable Transformation

New **re-usable transformations** are created from the **Tools > Define Re-Usable Transformations** menu.



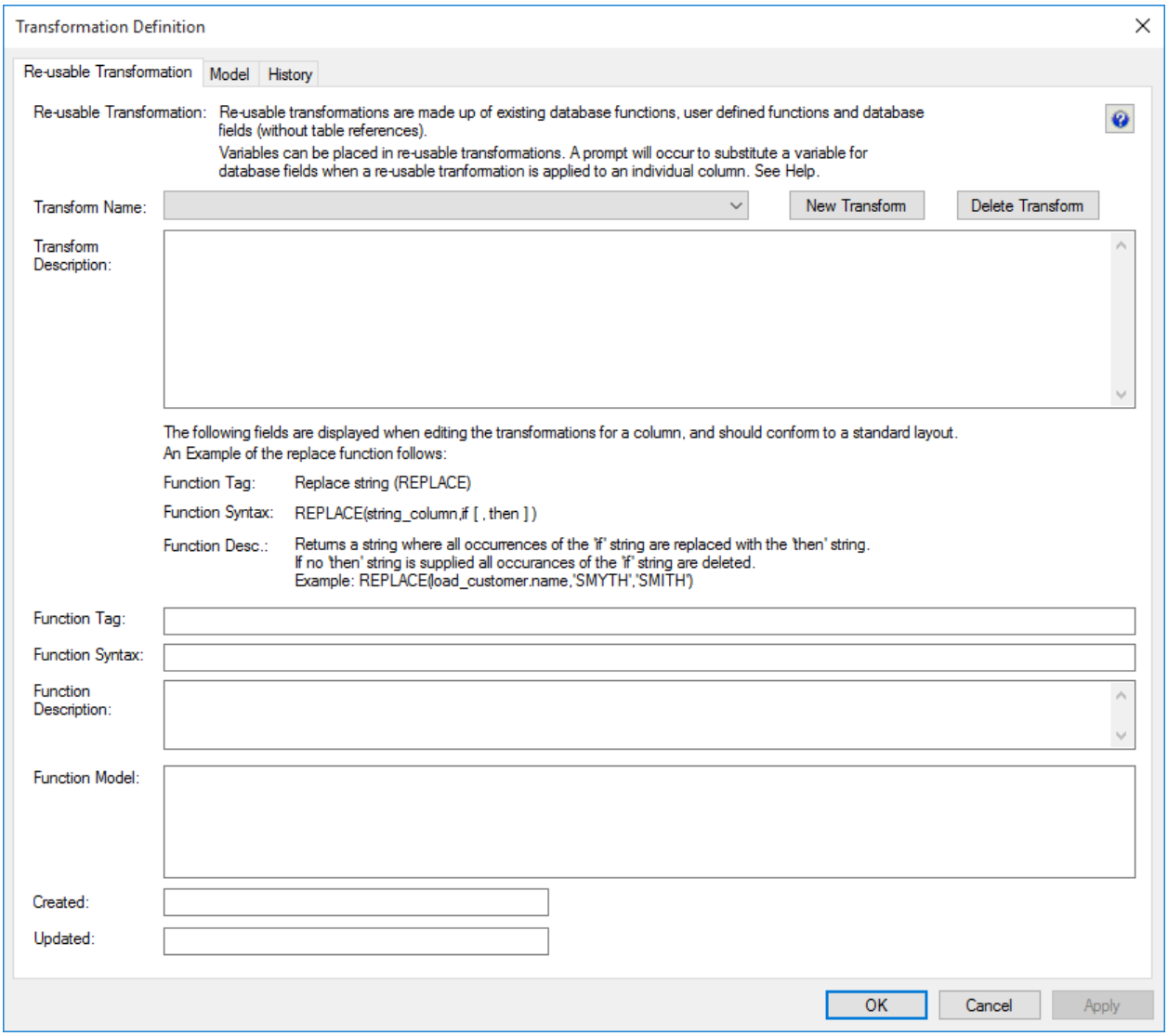
Creating a new re-usable transformation is a three step process:

1. Specify the name of the transformation
2. Enter metadata for the transformation
3. Define the transformation

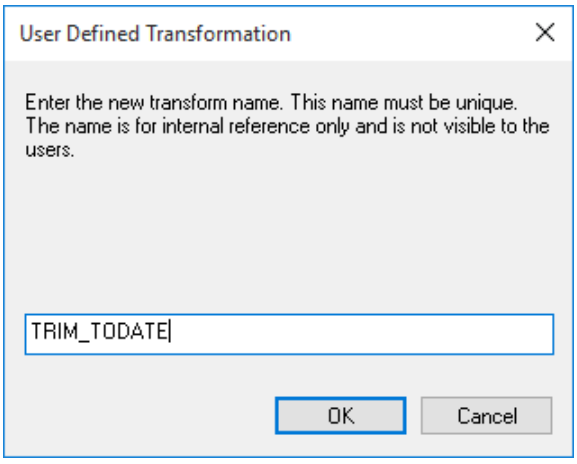
© Copyright, WhereScape Software Ltd All rights reserved.

Specify the Name of the Transformation

After selecting **Define Re-Usable Transformations** from the **Tools** menu, the following screen is displayed:



Click **New Transform** and enter a name for the re-usable transformation:



Note

This is the internal WhereScape RED name for the transformation, not the name developers reference when utilizing the transformation on column transformations.

Click **OK**.

© Copyright, WhereScape Software Ltd All rights reserved.

Enter Re-usable Transformation Metadata

Enter the following metadata for the transformation to describe the transformation for developers.

Field	Description
Transform Description	A general description of the transformation.
Function Tag	This is the function name visible to the users, which can be selected from the function builder when building column transformations.
Function Syntax	The syntax guide for the function. This is visible in the function builder when clicking on the User defined function.
Function Description	The description of the function visible in the function builder when clicking on the User defined function.
Function Model	The model (template SQL code) for a User Defined Transformations. This is only visible for User Defined Transformations. [Read only].

© Copyright, WhereScape Software Ltd All rights reserved.

Define the Transformation Model

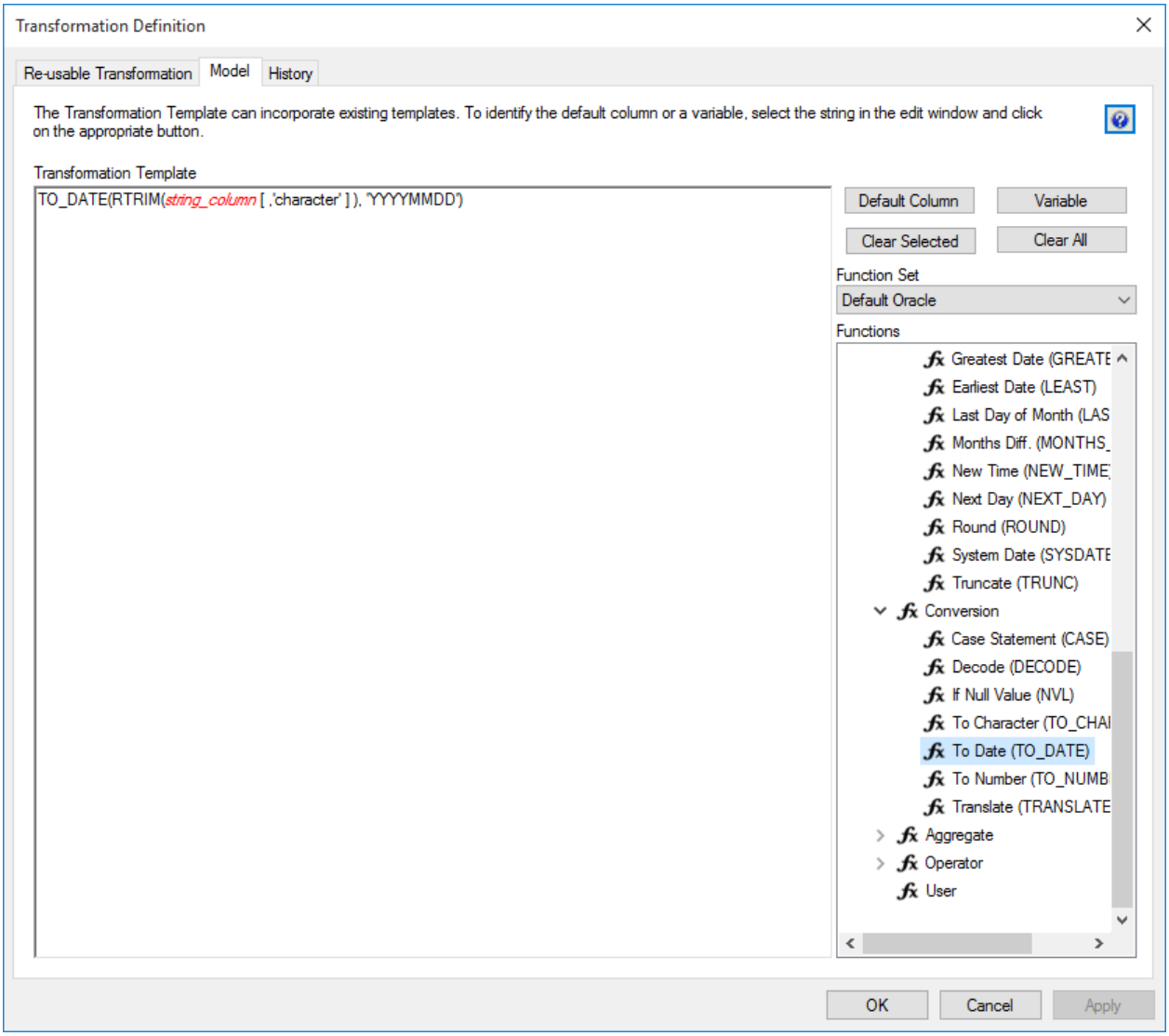
Once the transformation has been created and the metadata entered, the actual SQL code used by the transformation can be defined. The SQL code can be entered directly or via the **Function Builder** on the right. To use the **Function Builder**, expand the tree to find the required function.

Oracle Example of Building a Transformation Model

To build a model TO_DATE, converting a trimmed string in YYYYMMDD format to a date, do the following:

1. Click the **Model** tab
2. Expand **Functions** heading
3. Expand the **Conversion** heading
4. Double-click **To Date (TO_DATE)**
5. Highlight **numeric_column**
6. Expand the **String** heading
7. Double-click **Right Trim (RTRIM)**
8. Highlight **[format]** and type **'YYYYMMDD'**

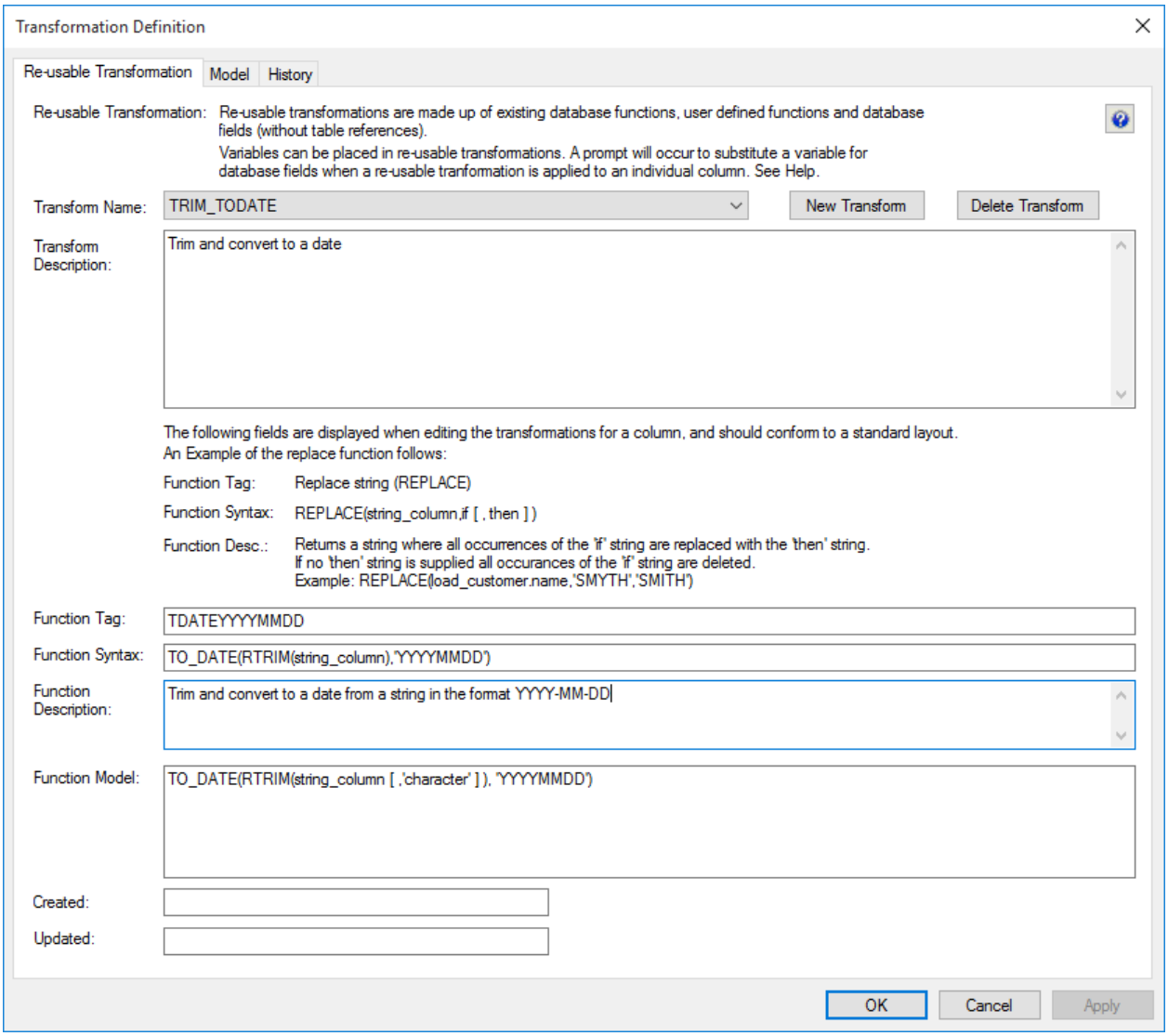
You should see the following:



Click **OK**.

© Copyright, WhereScape Software Ltd All rights reserved.

Completed Re-usable Transformation



Changing a Re-usable Transformation

To change a re-usable transformation:

1. Select **Re-Usable Transformations** from the **Tools** menu.
2. Choose the transformation from the **Transform Name** drop-down list.
3. Click on the **Model** Tab.
4. Change the SQL as required.
5. Click **OK.**

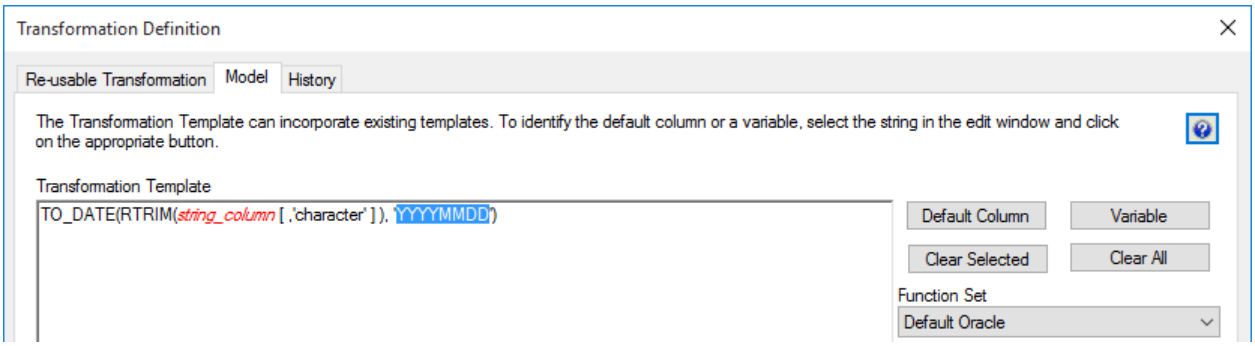
Example of a change to the Model SQL

In the example used in Creating a New Re-usable Transformation, the SQL was:

TO_DATE(RTRIM(string_column),'YYYYMMDD')

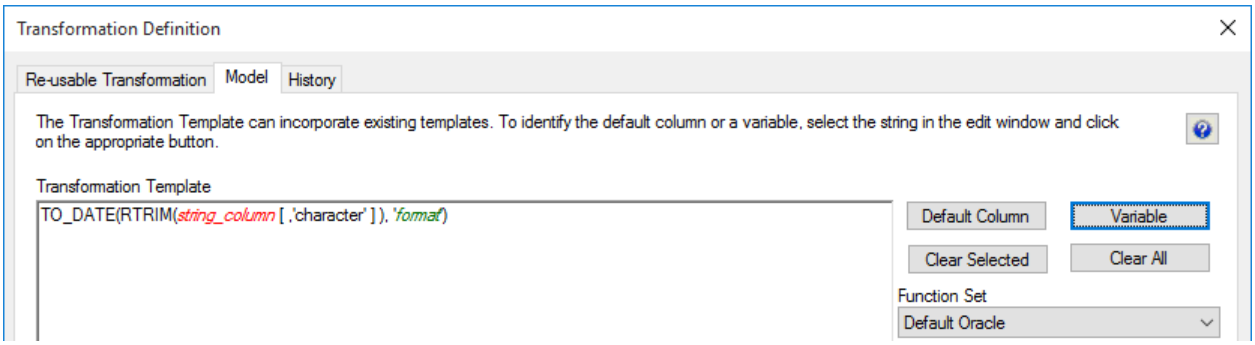
Change the SQL to allow the format to be specified when the transformation is used by changing **YYYYMMDD** to **format**.

© Copyright, WhereScape Software Ltd All rights reserved.



Then highlight the word **format** and click on the **Variable** button. This makes the word **format** a variable than can be substituted when the Re-usable Transformation is used.

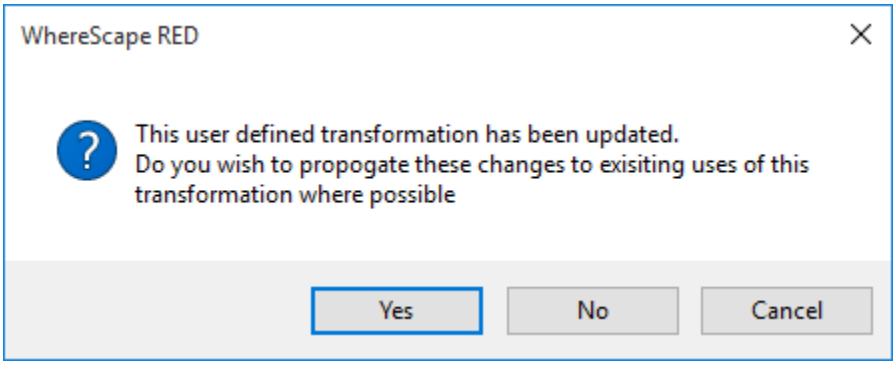
Now **format** is green and in italics:



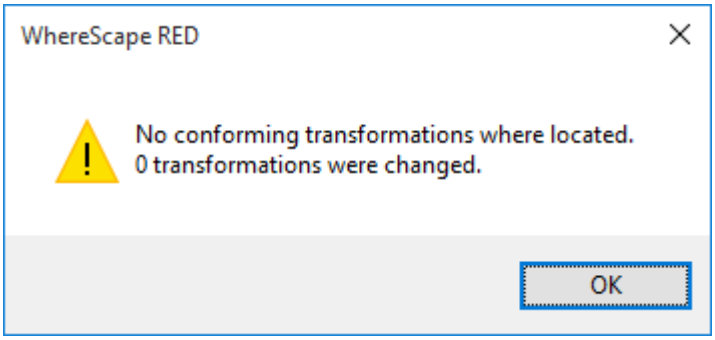
Click **OK**.

Applying Changes to Dependant Transformations

After changing a **Re-usable Transformation**, a dialog appears asking to confirm that changes should be applied to individual columns using the transformation, where possible:

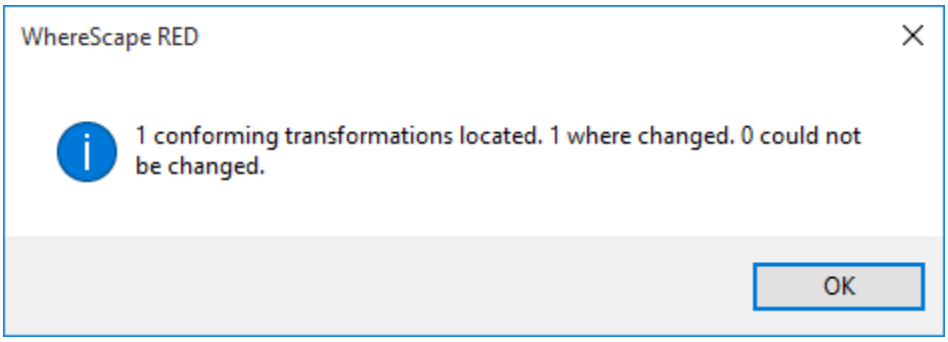


If the **Re-usable Transformation** doesn't have any dependant columns, then the following message is displayed:

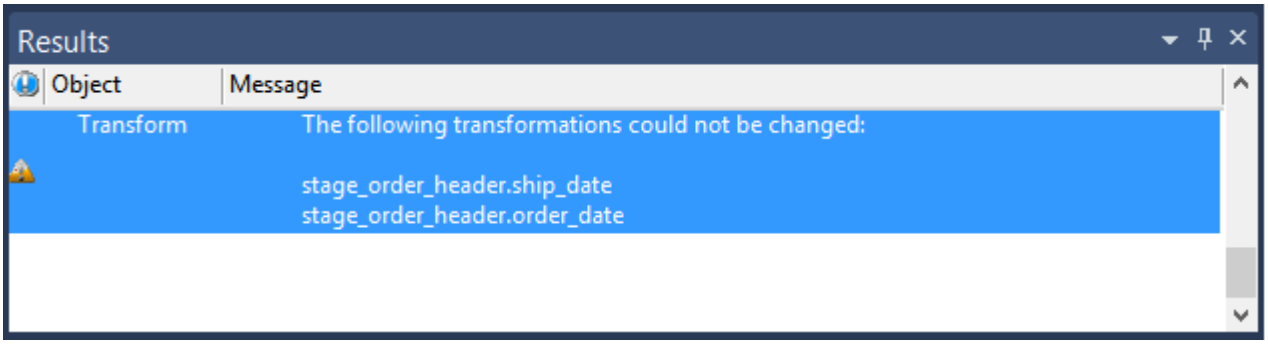


If the **Re-usable Transformation** has been used for dependant columns then this message is displayed:

© Copyright, WhereScape Software Ltd All rights reserved.



When an attempt is made to update a dependant Transformation, and the transformation has been modified in such a way as to make it impossible for the changes to the Re-usable Transformation to be applied, the error message above will include a count of the failures. The Results pane will detail the columns (and tables) where the update failures have occurred:



Using Re-usable Transformations

Re-usable transformations are used exactly the same way as any standard database **Function**. They can be used on any object type. Refer to **Column Transformation Properties** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

Procedures and Scripts

WhereScape RED has a **Procedure** object group for database stored procedures and a **Script** object group for host system scripts, such as UNIX shell scripts or Windows batch files.

Procedures

WhereScape RED generates the bulk of the procedures during a prototype build, but these procedures can be customized. In fact it would be normal practice once the prototype phase is completed to modify these procedures to meet specific requirements. The procedure object group refers to the concept of database stored procedures. Specific objects may in fact be functions, procedures or packages.

Scripts

Host scripts are generated when a script based file load is chosen during a file drag and drop from a Windows or UNIX connection. Scripts can also be created manually provided the rules for their inclusion into the scheduling process are followed.

Notes

- Windows script supports both DOS Batch and **PowerShell scripts**.
- While Windows and UNIX scripts are available on Load table objects, currently only Windows Scripts are available on other object types that are created in a Custom Database Connection.

This chapter covers the generation, editing and compilation of procedures, as well as the generation, editing and testing of host scripts as well as explaining the components required to allow host scripts to work in the scheduler environment.

Procedure Generation

WhereScape RED generates template procedures to assist in the various phases of the data warehouse build process. A procedure is generated by selecting the **(Build Procedure...)** option from a drop-down list field in a table's Properties window to configure the update, initial build, or post load procedures.

For **Load** tables, a post load procedure can be generated by selecting the option above. This post load procedure is designed to assist in the management of file loads where a trigger file has been used.

Post load procedures can either be manually generated from a RED provided procedure outline or generated leveraging a RED template—refer to **Rebuilding Update Procedures** for details.

Note

For custom database targets, the option to use Post Load Scripts is provided, as an alternative to post load procedure. Post load scripts can be generated using a RED template—refer to **Rebuilding Update Procedures** for details

For **Dimension**, **Stage**, **Fact** and **Aggregate** tables, an Update or Initial Build procedure can be generated by selecting from the corresponding procedure drop-down list field in the table's Properties.

When an 'Update' procedure generation is selected for a Dimension, a **get_key function** is also automatically created.

If a new procedure is created from scratch (i.e. not auto generated), then an outline of the syntax required by the WhereScape scheduler can be generated by selecting the **Tools > Create Procedure Outline** menu option in the procedure editor.

© Copyright, WhereScape Software Ltd All rights reserved.

Wrapper procedures

In some cases, multiple procedures will be required to update a table. In such cases, it is best to create a top level procedure that is seen by the scheduler as the 'Update' procedure. This procedure can in turn, call other procedures.

Example

We may have a fact table that is updated from multiple stage tables. This Oracle wrapper procedure calls two child procedures, one for each stage table that is to update the fact table. A status is reported back to the audit trail for each stage and an overall status ascertained for the fact table update.
The Oracle wrapper procedure may look as follows:

```
CREATE OR REPLACE procedure update_fact_sales_detail
    (
    v_sequence          IN  number,
    v_job_name          IN  varchar2,
    v_task_name         IN  varchar2,
    v_job_id            IN  number,
    v_task_id           IN  number,
    v_return_msg        OUT varchar2,
    v_status            OUT number
    )
    AS

    --------------------------------------------
    -- Control variables used in most programs
    --------------------------------------------
    v_MsgText          varchar2(255); -- Text for audit_trail
    v_step             integer := 0;  -- return code
    v_invoice_msg     varchar2(256);  -- from invoice update
    v_order_msg       varchar2(256);  -- from order update
    v_invoice_status number  := 0;    -- from invoice update
    v_order_status   number  := 0;    -- from order update
    v_retcode         varchar2(1);

BEGIN

    ----------------------------------------
    -- Update fact table with info from orders stage table
    -- Report the results to the audit trail.
    ----------------------------------------
    v_step := 10;
    update_fact_sales_detail_o(v_sequence,v_job_name,v_task_name,
                v_job_id, v_task_id,v_order_msg, v_order_status);
    If v_order_status = 1 Then
        v_retcode := 'S';
    ElsIf v_order_status = -1 Then
        v_retcode := 'W';
    ElsIf v_order_status = -2 Then
        v_retcode := 'E';
    ElsIf v_order_status = -3 Then
        v_retcode := 'F';
    End If;
    v_status := WsWrkAudit(v_retcode,v_job_name,v_task_name, v_sequence,
               v_order_msg,NULL,NULL,v_task_id, v_job_id);

    ----------------------------------------
    -- If The previous update (orders) was successful
    -- Update fact table with info from invoice stage table
    ----------------------------------------
    If v_order_status > -2 Then
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
        v_step := 20;
        update_fact_sales_detail_i(v_sequence,v_job_name,v_task_name,
                    v_job_id, v_task_id,v_invoice_msg, v_invoice_status);
        If v_invoice_status = 1 Then
            v_retcode := 'S';
        ElsIf v_invoice_status = -1 Then
            v_retcode := 'W';
        ElsIf v_invoice_status = -2 Then
            v_retcode := 'E';
        ElsIf v_invoice_status = -3 Then
            v_retcode := 'F';
        End If;
        v_status := WsWrkAudit(v_retcode,v_job_name,v_task_name, v_sequence,
                   v_invoice_msg,NULL,NULL,v_task_id, v_job_id);
    Else
        v_invoice_status := 0;
    End If;

    --------------------------------------
    -- All Done report the results
    -- If any of our called procedures failed or gave
    -- warnings then report the final status as such
    --------------------------------------
    v_step := 90;
    v_status := v_order_status;
    If v_invoice_status < v_order_status Then
        v_status := v_invoice_status;
    Else
        v_status := v_order_status;
    End If;
    v_return_msg := 'fact_sales_detail update ';
    If v_status = 1 Then
        v_return_msg := v_return_msg||'completed.';
    ElsIf v_status = -1 Then
        v_return_msg := v_return_msg||'completed with warnings.';
    Else
        v_return_msg := v_return_msg||'Failed.';
    End If;

    RETURN;

EXCEPTION
  WHEN OTHERS THEN
    v_return_msg := 'Unhandled Exception in '||'update_fact_sales_detail. '||
                ' Step ' ||v_step || ' '||SQLERRM;
    v_status := WsWrkAudit('F',v_job_name,v_task_name, v_sequence,
                v_return_msg,SQLCODE,SQLERRM,v_task_id, v_job_id);
    v_status := -3;

    RETURN;
END update_fact_sales_detail;
```

© Copyright, WhereScape Software Ltd All rights reserved.

Procedure Placeholders

Procedure placeholders can help in moving procedures between environments without the necessity of regenerating those same procedures. In RED, the purpose of these placeholders is to automatically substitute the corresponding strings, which is needed for a specific environment.

The following procedure placeholders described below can be found in the update_xxxx_xxxx procedure.

[TABLEOWNER] is used as a placeholder to replace the schema name defined in the connection or target.

For targets, the **[TABLEOWNER]** placeholder is derived from the Target Location Database/Schema in the connection. The target can be changed in the table's **Storage** screen, on the **Target Location** drop-down list. For more information about Target Location Database/Schema in connections and table storage screens, refer to **Connections - Database**, **Connections - ODBC** and **Table Storage Properties** for details.
When moving tables between environments, the **[TABLEOWNER]** placeholder is determined by the individual connection of the target environment.

Example

During the compilation process of the procedure, [TABLEOWNER].[table_name] is replaced with PRODUSER.tablename, if the table owner is PRODUSER in the destination environment.

```
-- ============================================================================
-- DBMS Name       :    Oracle
-- Script Name     :    update_dim_customer
-- Description     :    Update the dimension dim_customer
-- Generated by    :    WhereScape RED Version 6.8.4.4 ( Build 151111-200341 pre-release )
-- Generated for   :    RED Documentation
-- Generated on    :    Tuesday, November 17, 2015 at 09:05:24
-- Author          :    WhereScape Documentation
-- ============================================================================
-- Notes / History
--

CREATE OR REPLACE PROCEDURE update_dim_customer
  (
  p_sequence         IN  number,
  p_job_name         IN  varchar2,
  p_task_name        IN  varchar2,
  p_job_id           IN  number,
  p_task_id          IN  number,
  p_return_msg       OUT varchar2,
  p_status           OUT number
  )
  AS

  --------------------------------------------------
  -- Control variables used in most programs
  --------------------------------------------------
  v_msgtext          varchar2(255); -- Text for audit_trail
  v_sql              varchar2(255); -- Text for SQL statements
  v_set              integer := 0;  -- commit set
  v_analyze_flag     integer := 0;  -- analyze flag
  v_step             integer := 0;  -- return code
  v_update_count     integer := 0;  -- no of records updated
  v_insert_count     integer := 0;  -- no of records inserted
  v_count            integer := 0;  -- General counter;

  --------------------------------------------------
  -- Variables
  --------------------------------------------------
  v_DimRec           [TABLEOWNER].[dim_customer]%ROWTYPE;

  --------------------------------------------------
  -- Select all input columns for all dimension rows
  --------------------------------------------------
```

Tip

WhereScape RED Tip: dim_date
The TABLEOWNER placeholder is especially useful in update procedures when the related table is moved to a different schema or environment. For example, when moving dim_date to other schemas, [TABLEOWNER] will be replaced with the schema of the table when the procedure is compiled.
To do this, prefix the table name in the procedure to [TABLEOWNER].[table_name] (e.g. [TABLEOWNER].[dim_date]). It is then only necessary to do a recompile, instead of rebuilding or

© Copyright, WhereScape Software Ltd All rights reserved.

Tip

regenerating the procedure.

Note

For Oracle and DB2 databases [TABLEOWNER] has already been added to the sample dim_date update procedures.

[FUNCTIONOWNER] is used as a placeholder for **Greenplum** databases to enable the creation of procedures in different targets without the need of procedure regeneration.

```
-------------------------------------------------------------------------------
-- DBMS Name       :    Greenplum
-- Script Name     :    update_stage_order_header_gp
-- Description     :    Update the Stage  table stage_order_header_gp
-- Generated by    :    WhereScape RED Version 6.8.4.4 ( Build 151111-200341 pre-release )
-- Generated for   :    RED Documentation
-- Generated on    :    Tuesday, November 17, 2015 at 09:46:12
-- Author          :    WhereScape Documentation
-------------------------------------------------------------------------------
-- Notes / History
--

CREATE OR REPLACE FUNCTION [FUNCTIONOWNER].[update_stage_order_header_gp]
  (
  p_sequence       IN    integer,
  p_job_name       IN    varchar(256),
  p_task_name      IN    varchar(256),
  p_job_id         IN    integer,
  p_task_id        IN    integer,
  p_parameters     INOUT varchar[][],
  p_return_msg     INOUT varchar(256),
  p_status         INOUT integer
  )
RETURNS RECORD
AS $BODY$
DECLARE
  ---------------------------------------------------------------------------
  -- Control variables used in most programs
  ---------------------------------------------------------------------------
```

[SEQUENCE.tablename] is used as a placeholder for **Oracle** and **Greenplum** database sequences where the Sequence name can be defined for artificial keys in a table's storage screen.

This placeholder can be edited in the table's **Storage** screen, on the **Artificial Key Sequence Name** field, if that table contains an artificial key. For more information about replacing the sequence name in Oracle, refer to **Table Storage Screen - Oracle** for details.

```
-------------------------------------------------------
-- If not found then add it if we have the flag set.
-------------------------------------------------------
WHEN NO_DATA_FOUND THEN
  IF p_auto_add = 'Y' THEN
    INSERT INTO [TABLEOWNER].[dim_customer]
    (
      dim_customer_key,
      customer_code,
      dss_update_time
    )
    VALUES
    (
      [SEQUENCE.dim_customer].NEXTVAL,
      p_customer_code,
      SYSDATE
    )
    RETURNING dim_customer_key INTO p_dim_customer_key;
```

© Copyright, WhereScape Software Ltd All rights reserved.

[SCHEMA.tablename] is used as a placeholder for **Oracle** database schemas in partitioned fact tables, to facilitate moving partitioned fact table procedures between environments without having to regenerate the procedures.

```
BEGIN

  --------------------------------------------------------
  -- Get the current partition tablespace in case we need
  -- to add new partitions
  --------------------------------------------------------
  v_step := 100;
  SELECT MAX(tablespace_name) INTO v_part_tablespace
  FROM   all_tab_partitions
  WHERE  UPPER(table_owner) = UPPER('[SCHEMA.fact_customer]')
  AND    UPPER(table_name) = UPPER('fact_customer');

  --------------------------------------------------------
  -- Get the first partition name and maximum value as we
  -- will place all prior periods in this partition
  --------------------------------------------------------
  v_step := 110;
  SELECT partition_name, high_value
  INTO   v_min_part, v_min_high_value
  FROM   all_tab_partitions
  WHERE  UPPER(table_owner) = UPPER('[SCHEMA.fact_customer]')
  AND    UPPER(table_name) = UPPER('fact_customer')
  AND    partition_position = (
    SELECT MIN(partition_position)
    FROM   all_tab_partitions
    WHERE  UPPER(table_owner) = UPPER('[SCHEMA.fact_customer]')
    AND    UPPER(table_name) = UPPER('fact_customer'));

  -- Get the part high value and period
  v_min_part_value := TO_NUMBER(v_min_high_value);
  v_min_part_period  := TO_NUMBER(SUBSTR(v_min_part,3,8));

  --------------------------------------------------------
  -- Loop through each partition we need to modify
  --------------------------------------------------------
```

Procedure Editing

WhereScape RED includes a procedure editor which allows the maintenance of the various procedures, functions and packages within the data warehouse. The editor is invoked by double-clicking on a procedure name in the left pane or by right-clicking on the procedure name and selecting **Edit the Procedure**.

This section discuss some of the features of the procedure editor which includes the following:

- SQL syntax highlighting
- improved find feature (repeated find, up and down, etc.)
- toggle the display of line numbering
- a status bar that includes line number, line length, selected text length, etc.
- a context sensitive toolbar
- tab to space conversion and vice versa
- uppercase, lowercase, titlecase text conversion command
- increase/decrease test indent command
- bookmark support

A procedure can be compiled by selecting the **Tools > Compile Procedures** menu option. Refer to **Procedure Compilation** for details.

In the following sections reference is made to a selected block of text. A selected block of text is a normal windows selection where the text in question is highlighted. Normally achieved by holding down the left mouse button and moving the cursor.

Indenting code

The **Tab** character inserts four spaces into the text. A **Shift+Tab** removes four spaces.

© Copyright, WhereScape Software Ltd All rights reserved.

Cut, Copy, Paste and Delete

The normal Windows cut, copy, paste and delete functions are available either through the toolbar or via the right mouse pop-up menu.

Indenting a block of text

A selected block of text can be indented by four spaces by depressing the **Tab** character. Each tab will indent by a further four spaces. A **Shift+Tab** removes four spaces from the front of each line in the selected block.

Commenting out a block of text

A selected block of text can be turned into a comment by right-clicking and selecting **Comment**. The editor places two dashes '--' at the front of each line in the selected block. In the same way a block of text can be uncommented by choosing the **Uncomment** option.

Note

Only lines that start with two dashes in the left most column can be uncommented.

Inserting Steps

The right-click menu option **Insert Step** inserts a code line of the format 'v_step := 1000;'. Each subsequent insert adds 100 to the step value. The **Alt+Z** key combination can also be used to insert a step line.

The v_step variable is used in the event of an unhandled exception. In such a case, the current v_step value is reported, and it may be possible to ascertain the code that failed based on the value of this v_step variable.

Note

If a step is inserted via this method, then the step numbering automatically resets for all steps numbering from 100 in increments of 100 when the procedure is compiled or saved.

Inserting Audit Message

The right-click menu option **Insert Audit Log Call** inserts several lines of code into the procedure which calls WsWrkAudit. This API is used to log messages to the audit log in the WhereScape RED scheduler. By default, the message is generated to log the current step number. It can be changed by altering the first added line. The **Alt+A** key combination can also be used instead of the right-click menu option.

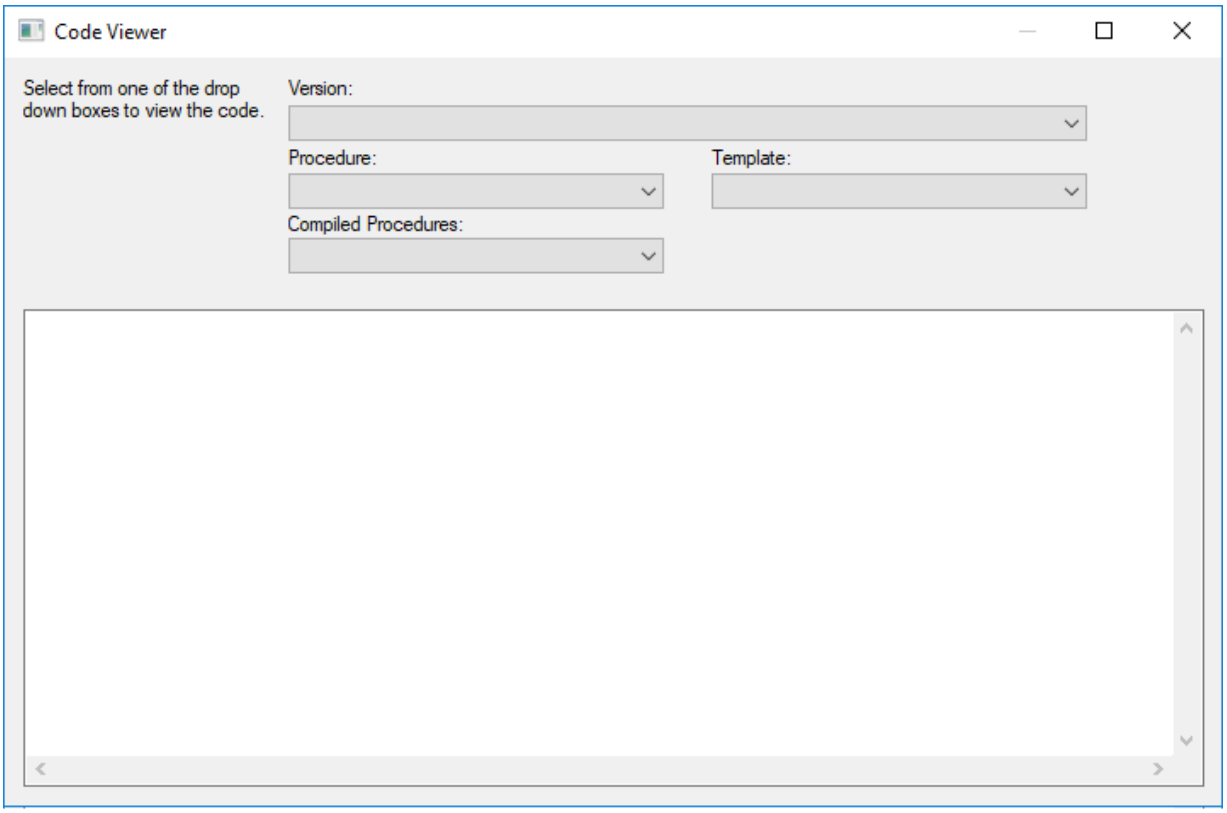
Inserting Detail/Error Message

The right-click menu option **Insert Detail/Error Log Call** inserts several lines of code into the procedure which calls WsWrkError. This API is used to log messages to the detail/error log in the WhereScape RED scheduler. By default, the message is generated to log the current step number. It can be changed by altering the first added line. The **Alt+D** key combination can also be used instead of the right-click menu option.

Viewing other procedural code

During the editing process it is possible to pop up a window containing other procedural code. This window allows cut and paste operations. In such a way it can be used as a work area or as a source of code. Select the **Tools > View Procedure or Template** menu option to bring the viewer window up. A dialog will appear as follows:

© Copyright, WhereScape Software Ltd All rights reserved.



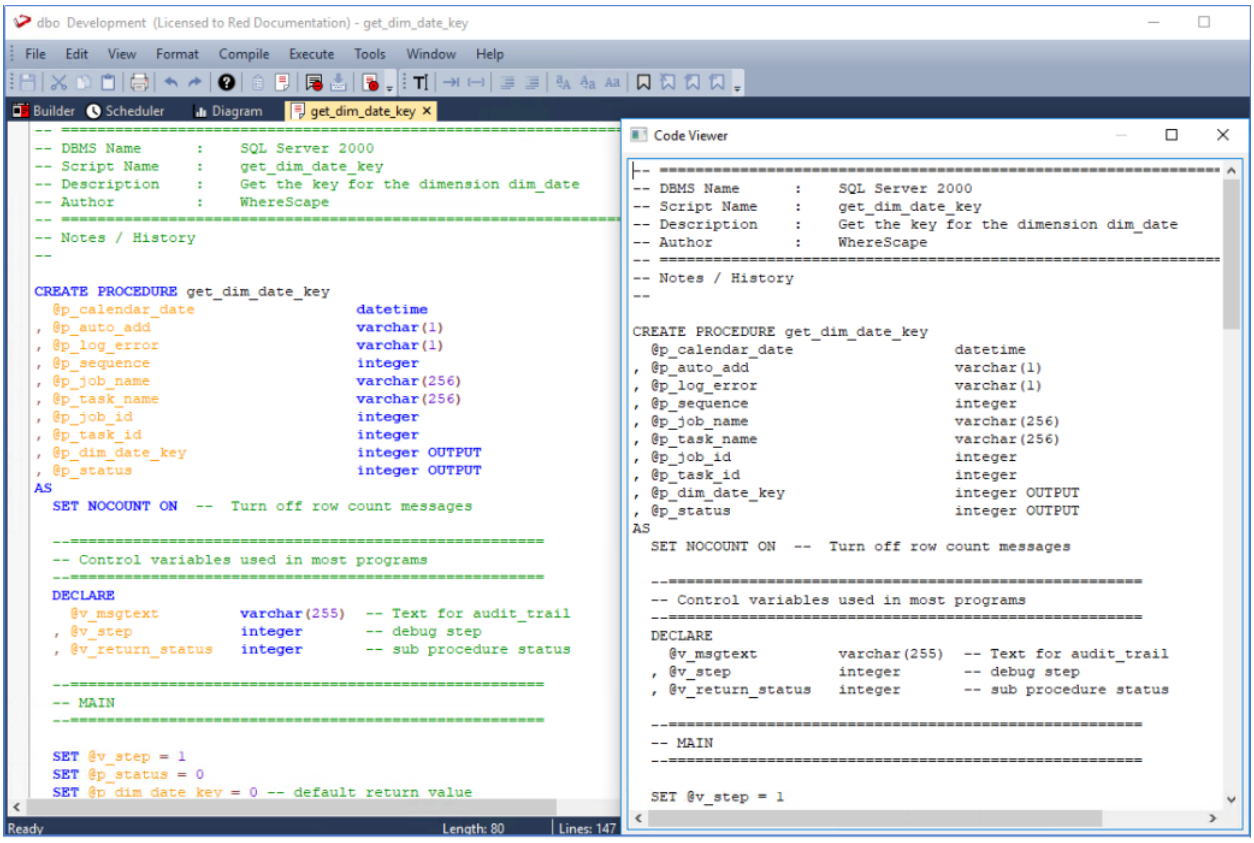
There are a number of drop-down list boxes to choose from. Once an item is selected, the viewer loads the code and moves to the right side of the edit window. The various options are:

- Version: A previously saved version of this procedure. The description and time are shown.
- Procedure: Some other procedure stored within the WhereScape RED metadata.
- Template: A template procedure as defined in the WhereScape RED metadata.
- Compiled procedure: One of the currently compiled procedures existing in the database.

Once an item is chosen, the viewer appears on the right side of the editor window. The viewer or the main editor window can receive the focus by clicking the mouse within the appropriate window.

Code can be cut from one window and pasted into the other. Any changes made in the viewer window cannot be saved. An example of an active view window is as follows:

© Copyright, WhereScape Software Ltd All rights reserved.



Note

Editing, deleting or compiling Locked for Edit or opened procedures/scripts

- Procedures or scripts cannot be deleted if they are Locked for Edit by any user, checked out by another user or if there is another object that has the same associated procedures or scripts.
- Saving or Compiling in the procedure or script edit window cannot be performed if the procedures or scripts become Locked for Edit by other users after the edit window has been opened.
- Procedures or scripts cannot be deleted or modified after the editor window has been opened, unless the Edit Lock has been released. Edit Locks can be released by any user in the Script or Procedure Properties screen.
- To prevent updates, deletes and modifications to certain procedures or scripts, it is best to use the Check Out functionality instead. For more information about this functionality, refer to **Object Check Outs and Check Ins** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

Procedure Loading and Saving

Procedures are normally stored in the WhereScape RED metadata tables. When a procedure is opened within WhereScape RED then the data is retrieved from the meta tables. Likewise when the procedure is saved, it overwrites the existing data in the meta tables.

When a procedure is compiled, it is also written to the database system tables. In the case of Oracle a compiled procedure can be viewed via the 'User_source' view (e.g. select text from user_source where name = 'MY_PROCEDURE' order by line;). For SQL Server the command is sp_helptext 'MY_PROCEDURE'.

Loading data

As mentioned above when a procedure is opened for editing the information stored in the metadata is loaded. Additional text can be loaded into the procedure edit window by selecting the **File > Insert from File** menu option which allows the browsing and inserting of a PC based file. Also paste buffer data can be inserted in the normal manner. In the previous 'Editing' section the viewer window was discussed. This window can also be a source of data via cut and paste.

Saving the Procedure

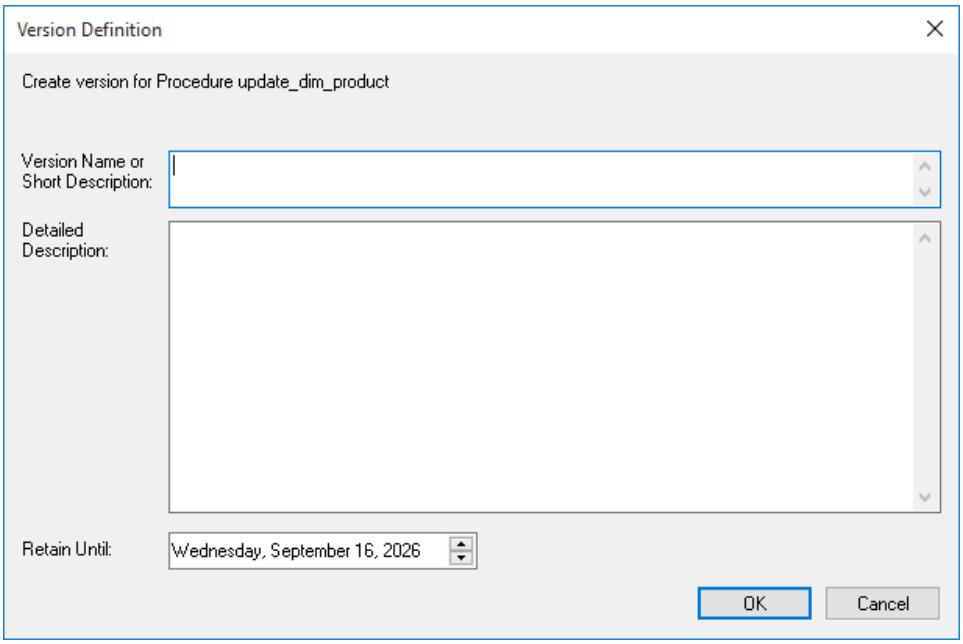
The default **File > Save Procedure** menu option overwrites the existing procedure in the metadata. In addition, a procedure can be saved to a Windows disk by selecting the **File > Write procedure to disk** menu option. All procedures can be written to disk from the main Builder menu option **Backup > Save procedures to disk**. This option allows the selection of a directory. The procedures are then written individually into that directory. All procedures are also written to one text file as part of the **Backup > Export the metadata** menu option.

Versions

Multiple versions of a procedure can be stored. Once a version is created that version may be read but may not be updated. Only the current procedure can be edited. There are a number of ways of creating a version of a procedure. These are:

1. By setting **Auto-Version before Procedure Compile** under **Tools > Options > Versioning.** If set to true, a new version of a procedure will be created whenever the procedure is compiled.
2. The Procedure Editor menu option **File > Save and Version** will save a procedure and create a version at the same time.
3. By selecting the **Version Control > New Version** menu option from the right-click context menu of a procedure in the main Builder window.
4. By selecting the **Tools > Version Objects** menu option.

When a version is created via method (2) or (3) above the following screen appears, to enable the definition of the version. If an auto version is created, then the person creating the version is recorded along with the reason for the version. (e.g. Version on compile, Version on procedure delete)



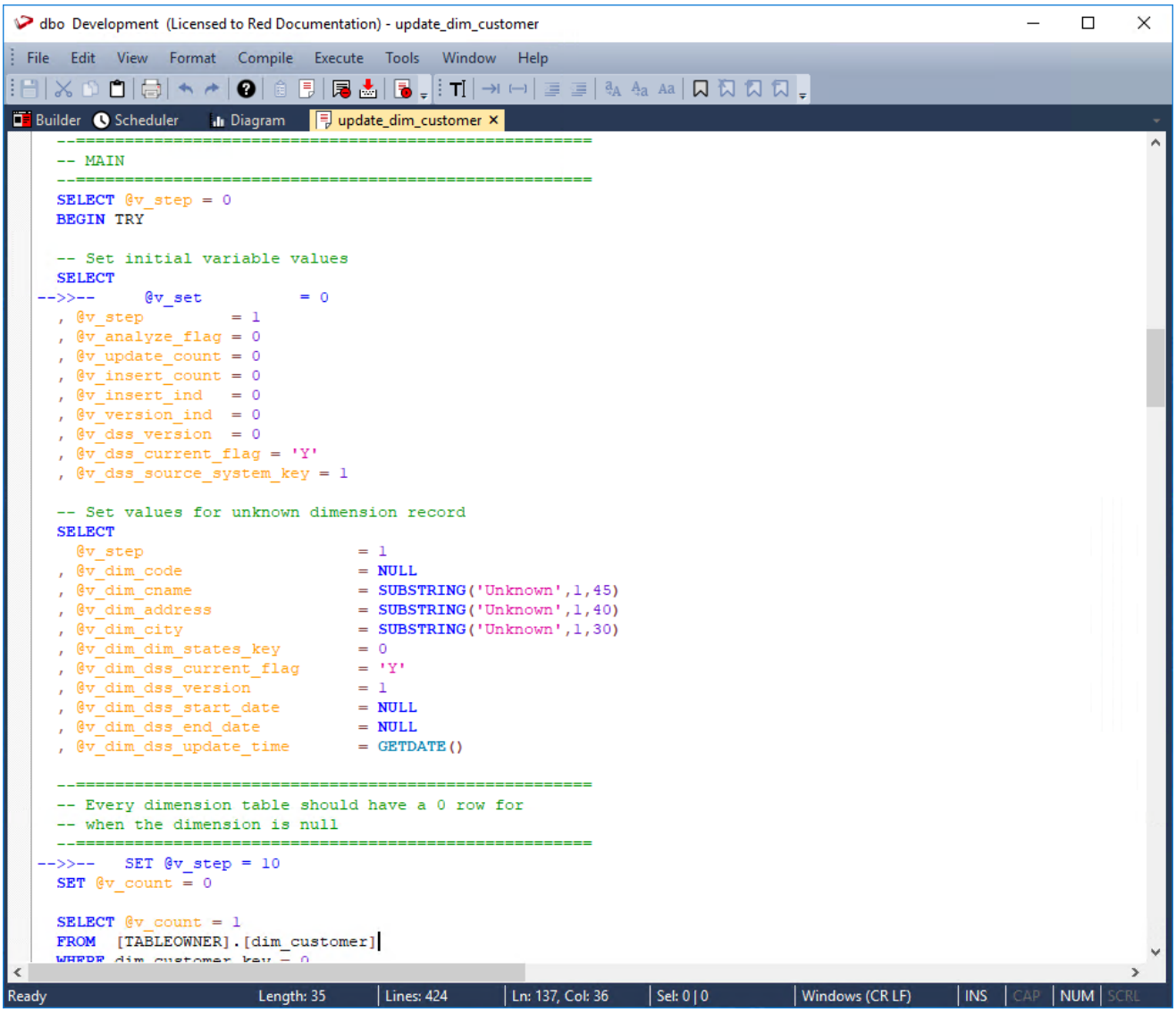
© Copyright, WhereScape Software Ltd All rights reserved.

- The version name/description appears when the versions are subsequently browsed.
- The Retain Until date is set ten years in the future by default.
- The automated deletion of versions is not supported.

Procedure Comparisons

A procedure can be compared to either an earlier version, or to the currently running code as compiled/stored in the database. The menu option **Tools > Compare to Compiled Source** enables the comparison of the procedure being edited with the code currently compiled and running in the database. If a viewer window is open (see the procedure editing section) then the **Tools > Compare to Viewer** menu option compares the contents of the viewer window with the current code. Therefore to compare against an older version, we first load the viewer window with the older version and perform a **Compare to Viewer**.

The comparison highlight the differences, as shown in the example below:



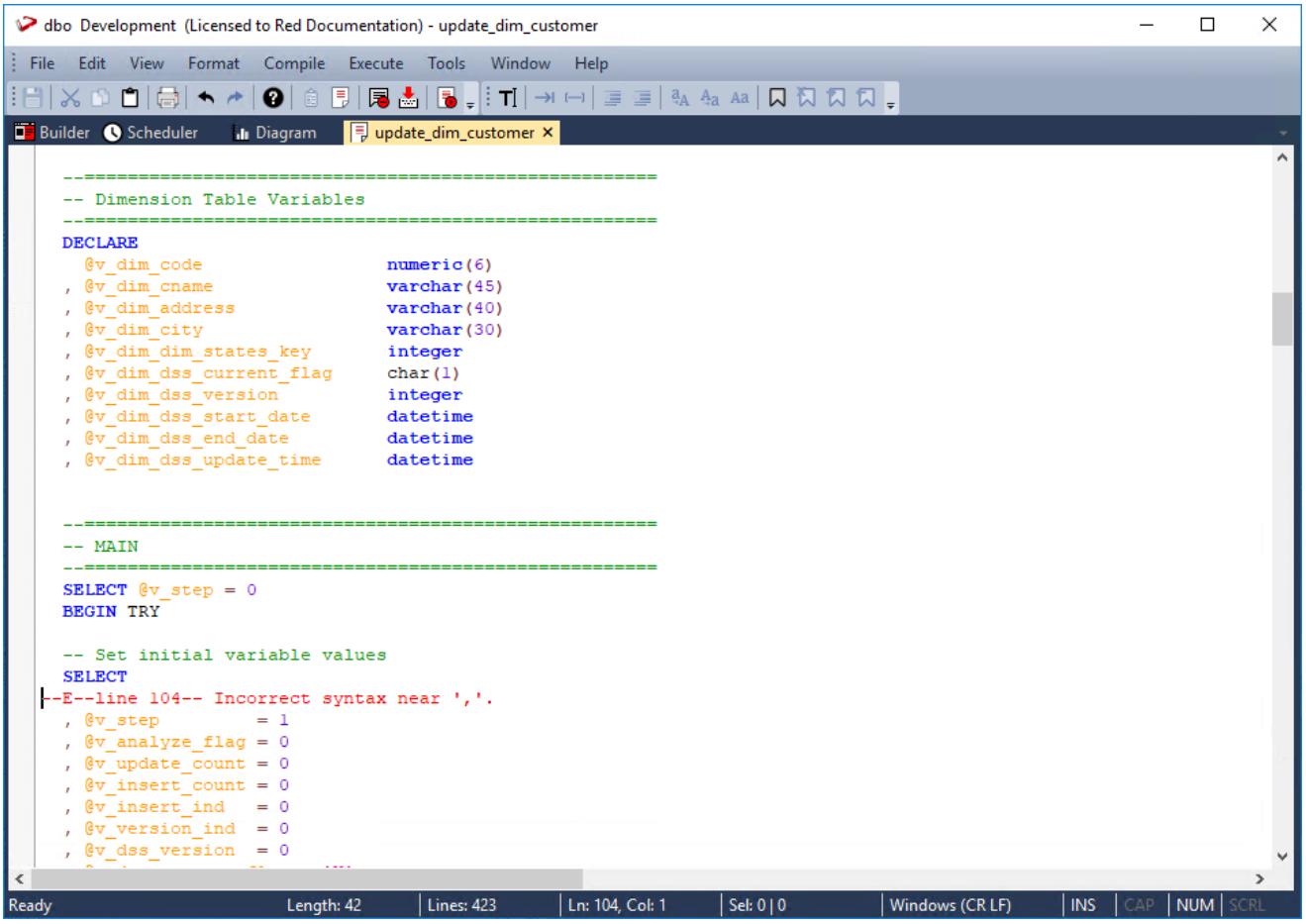
In this example, the lines **select @v_step = 0** and **set @v_step = 10** have been removed from the current code in the edit window.

Once the comparison has been completed, you can either remove the compare comments or accept the compare changes. The menu option **Tools > Remove Compare Comments** removes the added blue comments and code. The menu option **Tools > Accept Compare Changes** implements the changes highlighted. For the above example the line **select @v_step = 0** and **set @v_step = 10** would be added.

© Copyright, WhereScape Software Ltd All rights reserved.

Procedure Compilation

From within the procedure editor a procedure can be compiled by selecting the menu option **Compile > Compile** or by clicking the **Compile** icon. If the procedure compiles successfully, a window appears confirming a successful compile. If the compile fails, then error message comments are inserted into the procedure code. In the following example, the error messages are in color red and begin with --E--.



Error comments is inserted at each error point. **A compile will delete any previous error comments**. Error comments can also be removed through the menu option **Compile > Delete Error messages.**

Note

In some instances, the error comments may not be positioned on the correct line. This can occur as the result of one or more procedure lines being wrapped. Therefore, ensure the procedure editor window is maximized when dealing with compile errors.

If a procedure fails to compile then it is invalidated in the database, and will not run until successfully compiled.

© Copyright, WhereScape Software Ltd All rights reserved.

Procedure Running

Only procedures that conform to the WhereScape RED scheduler syntax can be executed from within the procedure editor. Select the **Execute > Execute** menu option or click the **Execute** icon to run the procedure. A procedure must have been compiled in order to run.

The results of the procedure is displayed in a window. The result code and result message is displayed, as well as any additional messages.

Procedure Syntax

The procedures managed by the WhereScape RED scheduler require the following standards. If a function or procedure is being developed that is not called directly by the scheduler, then it does not need to conform with this standard. If however, such a procedure or function wants to log messages to the audit or error logs then it needs the input parameters included in its parameter list.

Note

The WhereScape RED procedure editor includes support for syntax highlighting.

Parameters

The procedure must have the following parameters in the following order:

Parameter name	Input/Output	Oracle Type	SQL Server/DB2 Type
p_sequence	Input	Number	Integer
p_job_name	Input	Varchar2	Varchar(256)
p_task_name	Input	Varchar2	Varchar(256)
p_job_id	Input	Number	Integer
p_task_id	Input	Number	Integer
p_return_msg	Output	Varchar2	Varchar(256)
p_status	Output	Number	Integer

The input parameters are passed to the procedure by the scheduler. If the procedure is called outside the scheduler then the normal practice is to pass zero (0) in the sequence, job_id and task_id. A description of the run can be passed in the job name and the task name is typically the name of the procedure.

The output parameters must be populated by the procedure on completion. The return_msg can be any string up to 256 characters long that describes the result of the procedures execution. The status must be one of the following values:

Status	Meaning	Description
1	Success	Normal completion
-1	Warning	Completion with warnings
-2	Error	Hold subsequent tasks dependent on this task
-3	Fatal Error	Hold all subsequent tasks

Note

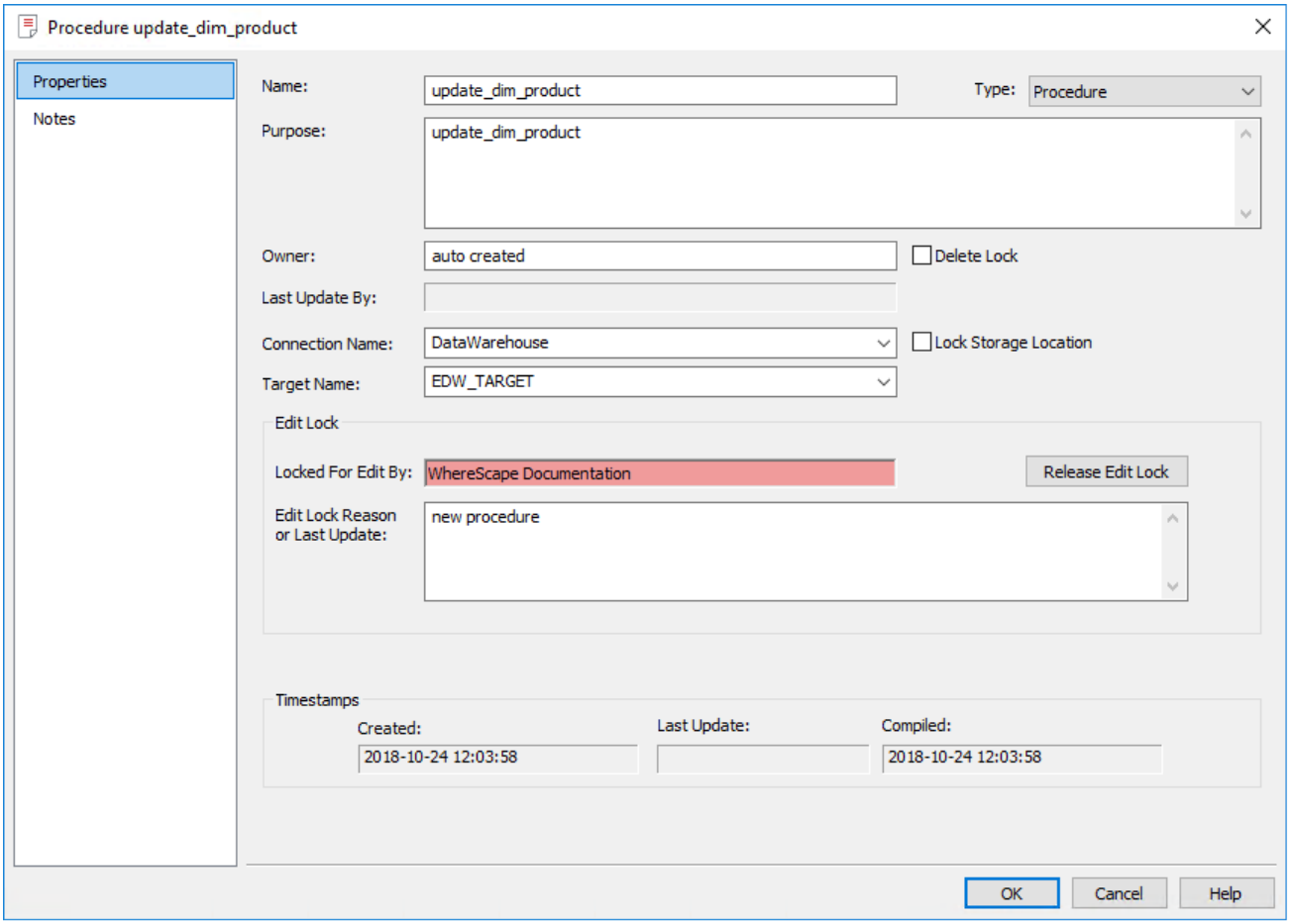
Multiple SQL statements can be separated using the "end of statement" indicator. This is <EOS> by default but can be configured **in Tools > Options > Code Generation > General**.

© Copyright, WhereScape Software Ltd All rights reserved.

Procedure Properties

The **Properties** screen for procedures and scripts is the same. A procedure can be renamed by changing the name field.

If a procedure is renamed, then it is also necessary to change the procedure name within the actual code. The **Purpose** and **Owner** fields are purely informational.



In the above example, the **Delete Lock** option is not selected. Selecting this check box prevents the procedure from being deleted through the **Delete** menu option. It also prevents the procedure from being overwritten, if a new procedure generation is requested.

Selecting the **Lock Storage Location** option enables you to retain the current location of the procedure, e.g. no changes will occur if the Target location of the procedure is manually changed, and a **Rebuild** or **Regenerate** operation is performed via the Table's **Properties** window or from the Table's right click **Code** context menu.

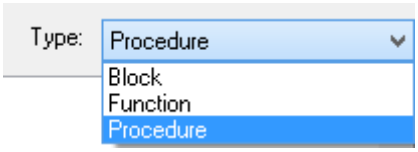
In the above example, the procedure is currently being edited, and is shown as **Locked for Edit** by "WhereScape Documentation". If procedures or scripts have already been opened for editing, they can only subsequently be opened for viewing.

Check out and delete operations for procedures and scripts, as well as the regeneration and drop of procedures are not permitted; if the object is currently Locked For Edit by another user.

The **Release Edit Lock** button to the right of the edit lock message, enables the edit lock to be cleared. If WhereScape RED, the database or the PC crashes when a procedure is open, then the check out must be cleared through this screen.

The **Edit Lock Reason** is for information only, and can be used as another comment field if desired.

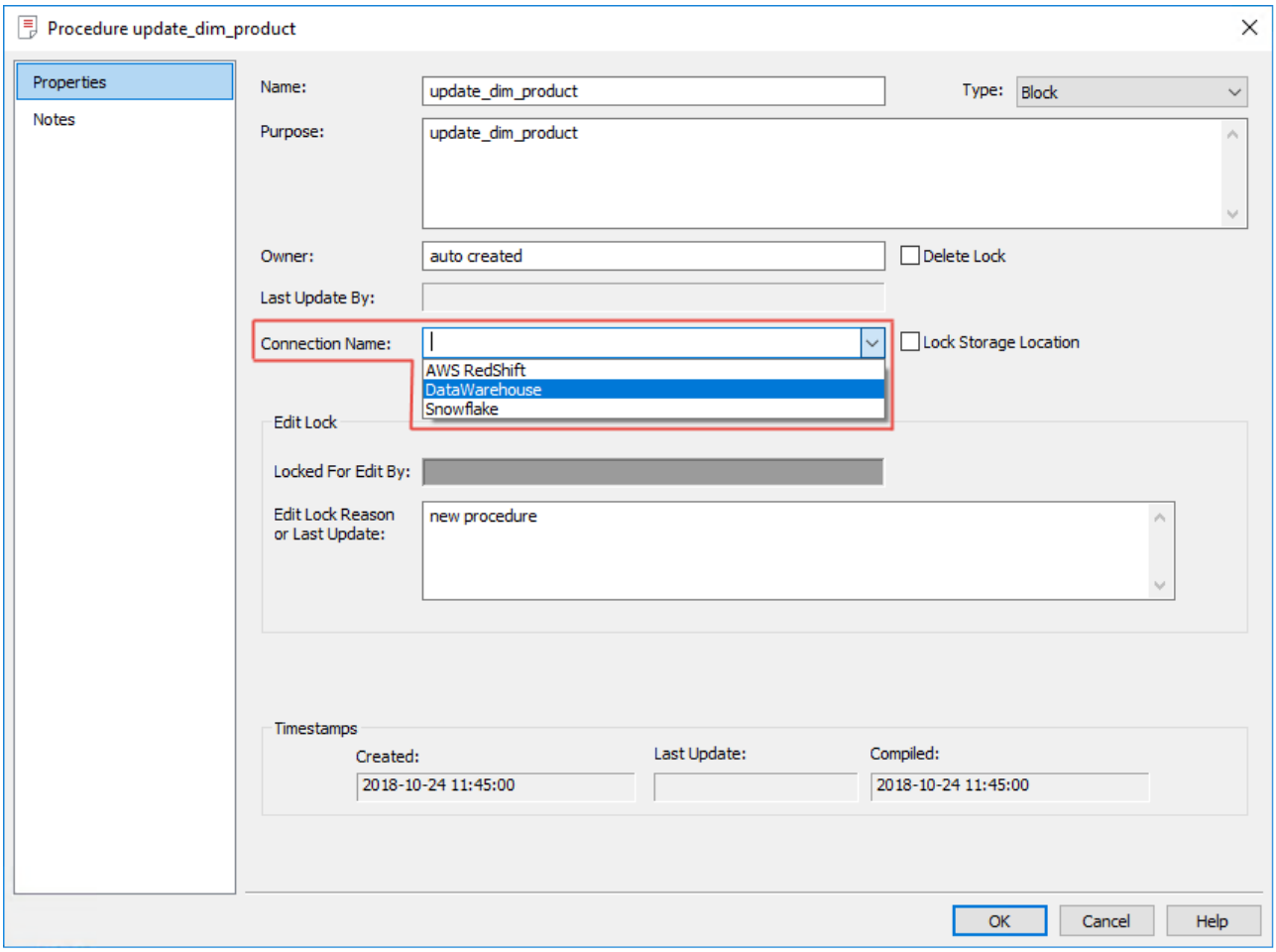
The **Type** drop-down list provides the selection of **Block**, **Function** or **Procedure**:



© Copyright, WhereScape Software Ltd All rights reserved.

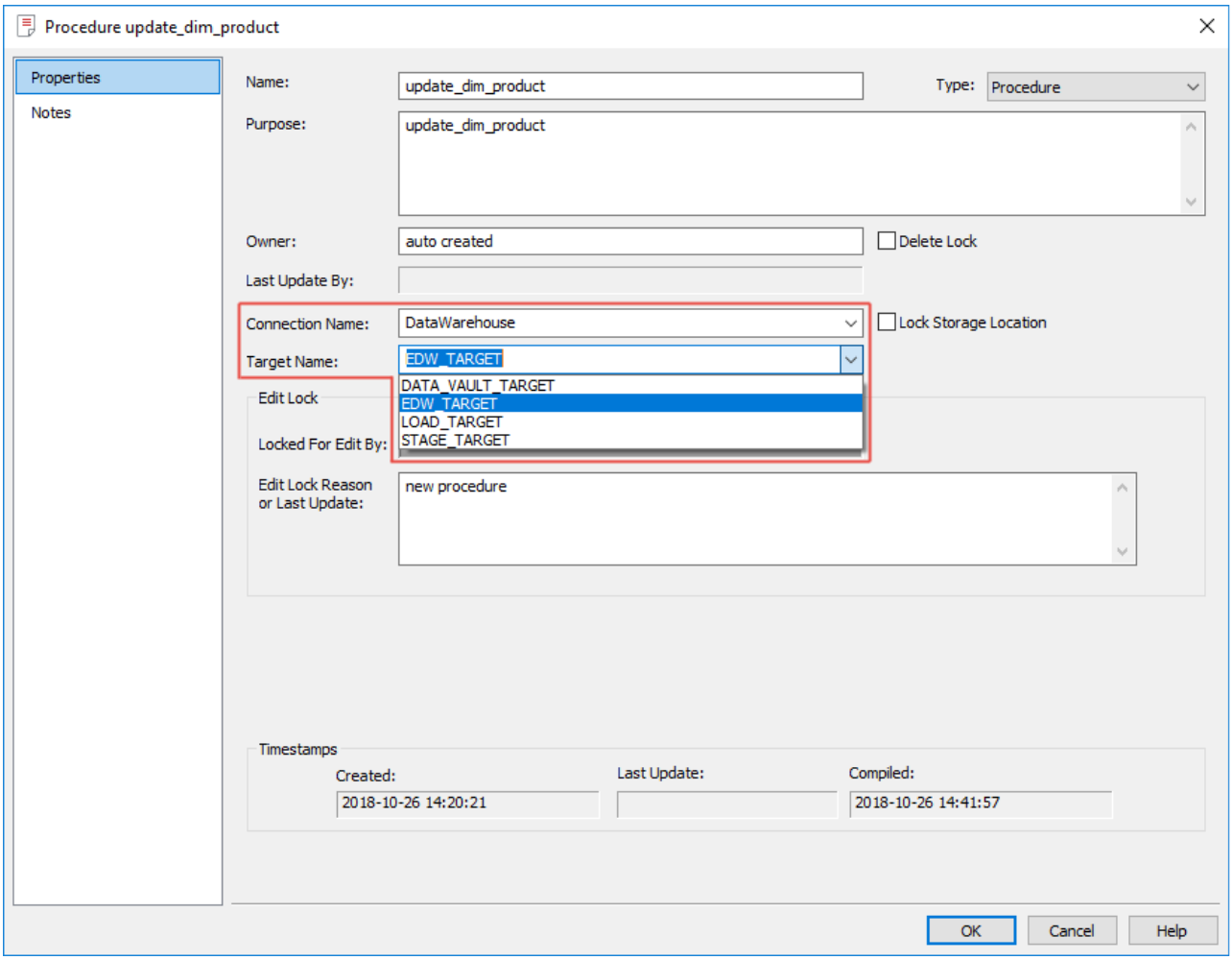
Selecting a type of **Block** enables you to execute a SQL block against another connection.

The **Connection Name** drop-down field enables you to select the connection against which the SQL block is executed.



For **Procedure** or **Function** type, the **Connection Name** drop-down field enables you to select the connection against which a procedure or function is executed and the target database where the update procedure or function is stored, via the **Target Name** drop-down field.

© Copyright, WhereScape Software Ltd All rights reserved.



Notes

- Target locations are defined, using the **Options > Storage > Target Location** setting. This setting enable users who are placing objects across multiple schema/database (in the same SQL Server instance) to set default target locations for new tables. The associated update procedures are stored in the RED repository meta database. However if required, the location of stored procedures can be changed and also be distributed across different Targets (e.g. same location as the created table).
- Selecting the **Lock Storage Location** option in the procedure **Properties** window, enables you to retain the current location of the procedure, if the Target location of the procedure is manually changed and a **Rebuild** or **Regenerate** operation is performed, via the Table object's **Properties** window or from the Table object's right click **Code** context menu
  Refer to **Procedure Migration** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

Procedure Migration

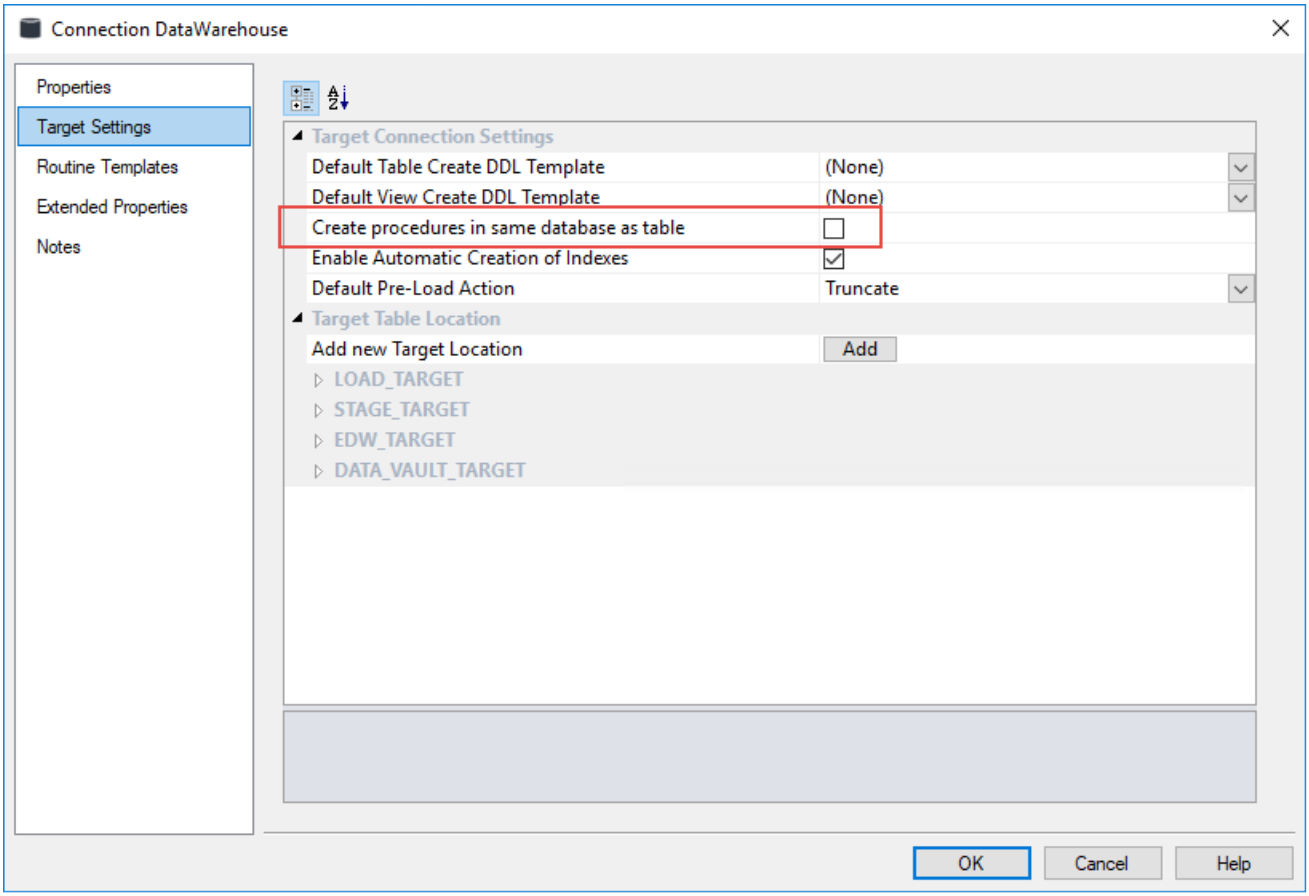
WhereScape RED provides the **Multiple SQL database** feature to enable tables to be distributed across multiple databases, instead of only a single database—refer to **Distributing Tables across Multiple SQL Server Databases** for details.

Update procedures for Table objects are stored in the RED repository meta database. However if required, the location of stored procedures can be changed and also be distributed across different Targets.

The following features in WhereScape RED are used for setting the location of stored procedures associated to a Table object in RED.

Connection Target Settings

The **Create procedures in same database as table** option in the **Target Settings** tab of the **Connection Properties** window, enables you to specify if the procedures associated with new Table objects are created in the same Target location as the Tables.



This option is not selected (OFF) by default, e.g. the table's stored procedures are created in the RED repository meta database.
If the Table is created in a different Target location (e.g. distinct schema/database in the same SQL Server instance), you can turn ON this option, if you want the stored procedures to also be located in the same Target.

Once this option is set, all new Table objects (created in the defined Target locations) will also have their generated procedures stored in the same Target as the Table objects.

Notes:

- This setting only applies for SQL server Target databases. For new RED repositories, this option is ON by default (e.g. procedures are created in the same location as the Table object). New or regenerated procedures are created in the specified location.
- Changing this setting has no impact on existing stored procedures. You must **Rebuild** or **Regenerate** the stored procedures for the changes to take effect.
- If the Target location of a Table object is changed, then both the **Re-creation** of the Table object (as in

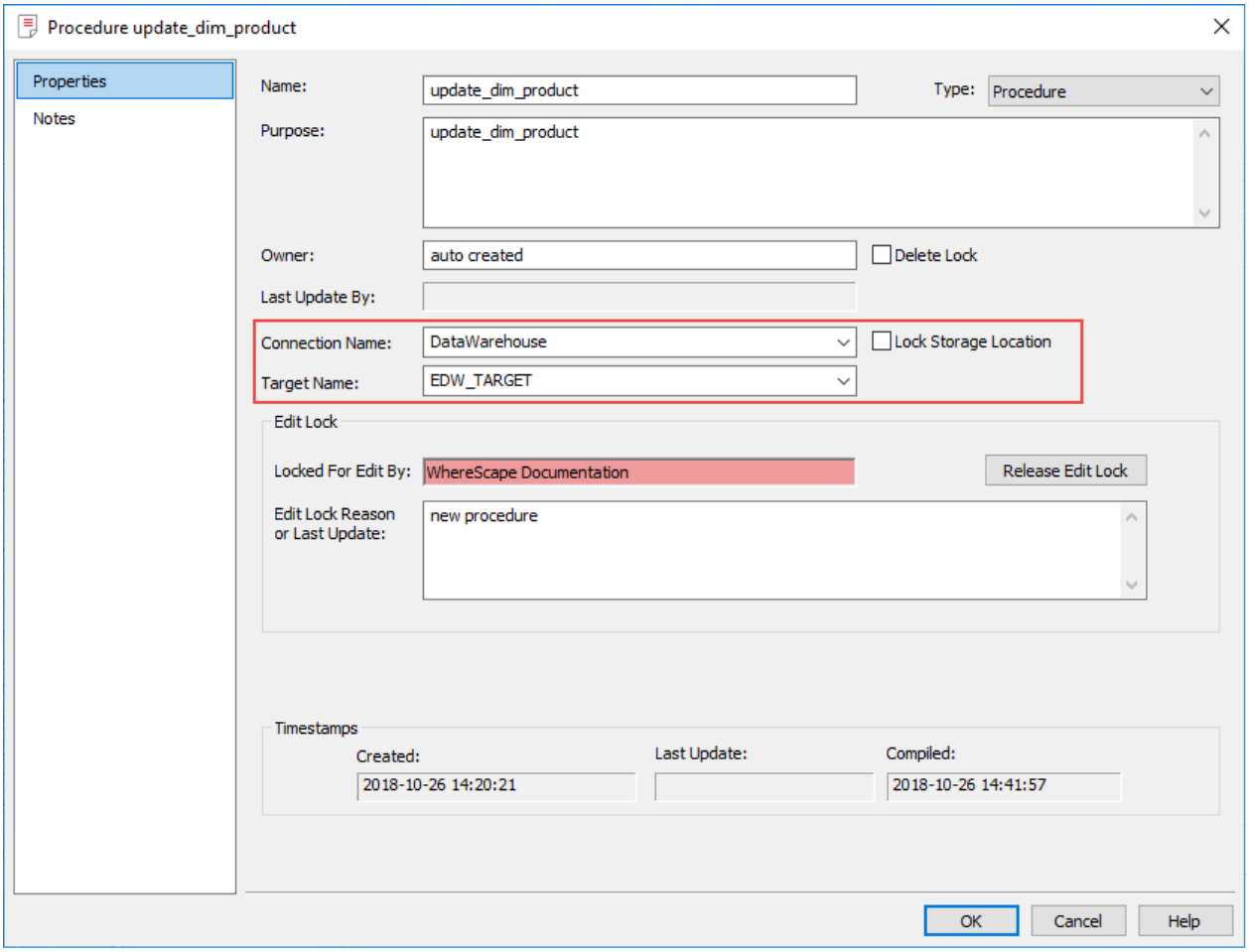
© Copyright, WhereScape Software Ltd All rights reserved.

Notes:

previous versions of RED) and the **Regeneration** of the Procedure object is required for both objects to be moved.

Procedure Properties

The **Connection Name** and **Target Name** fields in the **Procedure Properties** window, enables you to change the connection against which the Procedure or Function is executed and the Target where the update Procedure or Function is stored.



An option to **Lock the Storage Location** can also be set, if you want to retain the current location of the procedure, e.g. no changes will occur if the Target location of the procedure is manually changed and a **Rebuild** or **Regenerate** operation is performed via the Table's **Properties** window or from the Table's right click **Code** context menu.

Procedure Placeholders

The following procedure placeholders are used for moving procedures to the required target database location(s):

1. **[SCHEMA.procedure_name].procedure_name** = resolves to **schema.procedure_name** during the compilation process of the procedure.

   This placeholder is used in the CREATE PROCEDURE statement to resolve the schema location for the procedure.
2. [**PROCEDUREOWNER].[procedure_name]** = resolves to **database.schema.procedure_name** during the compilation process of the procedure.

© Copyright, WhereScape Software Ltd All rights reserved.

This placeholder is used to fully qualify procedures, such as GET_KEY_procedure from other databases.

3. **[METABASE].wsl_callable_routine** = resolves to **database.schema.wsl_callable_routine** during the compilation process of the procedure.

   This placeholder is used for each callable routine to always fully qualify to the metadata database and schema.

Moving Existing Procedures

The steps for moving modified and non modified procedures are outlined below:

Non-modified

In this case, an existing procedure generated by RED is being moved to another Target location.

1. After changing the Target location of the stored procedure, you must regenerate the procedure using the **Code > Regenerate update_procedure** option from the Table object's right click context menu.
2. Check the definition of the regenerated procedure to ensure that it is updated to include the required procedure placeholders and Table mark-ups.

Modified

In this case, a user created procedure or a modified RED procedure is being moved to another Target location.

Applying the procedure and table mark-ups in the procedure

1. Open the modified procedure via the procedure editor window.
2. Edit the procedure as follows:
   - Change the CREATE PROCEDURE syntax:

     CREATE PROCEDURE **[SCHEMA.procedure_name].procedure_name**
   - Ensure that all referenced procedures have the procedure owner mark-up:

     [PROCEDUREOWNER].[referenced_procedure_name]
   - Ensure that all source table references have the table owner mark-up:

     [TABLEOWNER].[table name]
   - Ensure all calls to metadata procedures and functions have the metadata database mark-up:

     Exec **[METABASE]**.WsWrkAudit
3. Save and compile any updates to the procedure.

Moving the procedure to the new database

1. Open the **Procedure Properties** of the modified procedure and set the **Connection Name** and **Target Name** fields to the appropriate connection and Target location for the stored procedure, and then click **OK**.
2. RED prompts if you want to **proceed with the change in storage location**, click **Yes** to proceed.

Note:

RED drops the procedure from the original database and creates it in the new Target location.

Repeat the same steps to move other modified procedures to another Target location.

Notes:

- When upgrading to version 8.3.1.0, the default connection and default target attributes for SQL Server procedure objects are cleared, to reflect that procedures are currently stored in the RED repository meta database.

© Copyright, WhereScape Software Ltd All rights reserved.

Notes:

- If the Target location of a Procedure is manually changed in the **Procedure Properties**, then regenerating or rebuilding the Procedure reverts its location back to the RED repository meta database or to the Table object's storage location (depending on whether the Target storage option is set in the **Connection Properties**).
- If you want to set the storage location of a Procedure to something other than your connection default Target, then set the **Lock Storage Location** option in the **Procedure Properties** to override the connection default Target.

Script Generation

WhereScape RED generates template scripts to assist in the loading of textual files from UNIX or Windows. These scripts are generated when a UNIX or Windows file is dragged into a Load table target and one of the 'Script based' file load options is chosen. Typically, script loads are used when some complexity prevents the use of the standard file based load. In such a case, the script will need modification.

Note

UNIX scripts are only supported for an Oracle data warehouse running on a UNIX platform.

The following sample scripts are covered in the remainder of this section:

1. Windows script for SQL
2. UNIX script for Oracle
3. Windows script for Oracle

Notes

- Windows script supports both DOS Batch and **PowerShell scripts**.
- While Windows and UNIX scripts are available on Load table objects, currently only Windows Scripts are available on other object types that are created in a Custom Database Connection.
- Additional host script language types which can be applied to Script objects can be defined in RED. Refer to **Host Script Languages** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

Script Generation (Windows/SQL Server)

A sample Windows script for SQL Server is as follows. The key components of the script are described below:

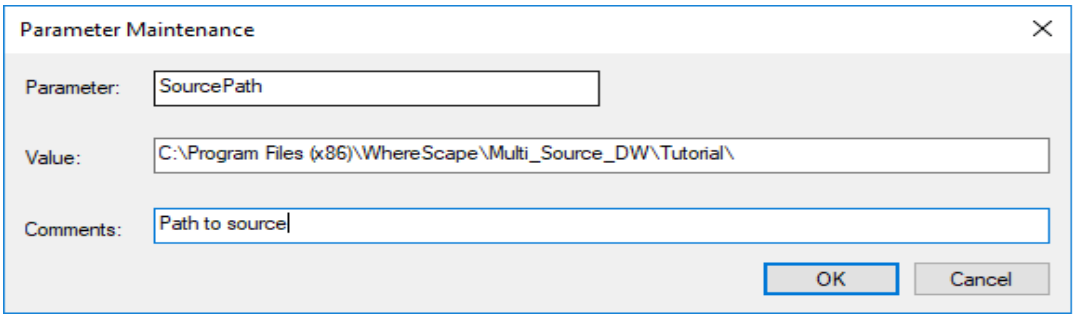
```
@echo off
setlocal enabledelayedexpansion
setlocal enableextensions
REM ************************************************************
REM ***** Load of C:\Program Files (x86)\WhereScape\Multi_Source_DW\Tutorial\forecast.txt into load_forecast
REM ************************************************************

REM ************************************************************
REM ***** NOTE: The following environment variables will be set
REM ***** WSL_LOAD_TABLE = the load table name
REM ***** WSL_LOAD_SCHEMA = the load table schema
REM ***** WSL_LOAD_DB = the load table database for the target connection
REM ***** WSL_SEQUENCE = a unique sequence number for the scheduler
REM ***** WSL_WORKDIR = the work directory defined in the connection
REM ***** WSL_BINDIR = the directory of the executable running this script
REM ***** WSL_TGT_SERVER = the server defined in the target connection
REM ***** WSL_DATABASE = the database defined in the connection
REM ***** WSL_USER = the dss user defined in the connection
REM ***** WSL_PWD = the dss password defined in the connection
REM ***** WSL_PARAMnnn = Any parameters that start with the load
REM ***** table name. For example: A table called load_abc has a
REM ***** parameter called load_abc_server defined. In this case
REM ***** a variable called WSL_PARAM_SERVER will be created
REM ************************************************************
REM If a wild card is used in the file name then %FILE_NAME%
REM in the control file must be enclosed in single quotes,
REM  otherwise it must not be enclosed in quotes.

SET LOAD_FILE="C:\Program Files (x86)\WhereScape\Multi_Source_DW\Tutorial\forecast.txt"
SET TRIG_FILE="C:\Program Files (x86)\WhereScape\Multi_Source_DW\Tutorial\forecast.trg"

if defined WSL_LOAD_TABLE (SET LOAD_TABLE=%WSL_LOAD_TABLE%) else SET LOAD_TABLE=load_forecast
if defined WSL_LOAD_SCHEMA (if "!WSL_LOAD_SCHEMA:~-1!" == "." (SET LOAD_SCHEMA=!WSL_LOAD_SCHEMA!) else SET LOAD_SCHEMA=!WSL_LOAD_SCHEMA!.) else SET LOAD_SCHEMA=dbo.
if defined WSL_LOAD_DB (SET LOAD_DB=%WSL_LOAD_DB%) else SET LOAD_DB=Multi_Source
if defined WSL_TGT_SERVER (SET SERVER=%WSL_TGT_SERVER%) else SET SERVER=DESKTOP-HGRS715
if defined WSL_DATABASE (SET DATABASE=%WSL_DATABASE%) else SET DATABASE=Multi_Source
if defined WSL_USER (SET USER=%WSL_USER%) else SET USER=
if defined WSL_USER (SET SQLCMDPASSWORD=!WSL_PWD!) else SET SQLCMDPASSWORD=
if defined WSL_WORKDIR (SET WORKDIR=%WSL_WORKDIR%) else goto LABEL_FILE_DEFAULT
if defined WSL_SEQUENCE (SET SEQUENCE=%WSL_SEQUENCE%) else goto LABEL_FILE_DEFAULT
SET FILECTL=%WORKDIR%\wsl%LOAD_TABLE%%SEQUENCE%.ctl
SET FILELOG=%WORKDIR%\wsl%LOAD_TABLE%%SEQUENCE%.log
SET FILEAUD=%WORKDIR%\wsl%LOAD_TABLE%%SEQUENCE%.aud
SET FILEFMT=%WORKDIR%\wsl%LOAD_TABLE%%SEQUENCE%.fmt
goto LABEL_WAIT
:LABEL_FILE_DEFAULT
SET FILECTL=c:\temp\wsl3.ctl
SET FILELOG=c:\temp\wsl3.log
SET FILEAUD=c:\temp\wsl3.aud
SET FILEFMT=c:\temp\wsl3.fmt
:LABEL_WAIT
```

Tip

WhereScape RED TIP: Parameters

Parameters can also be added to Scripts to facilitate deployment processes or environment changes without the need to regenerate scripts. Parameters can be added to scripts of Load and Export tables.
For example: add $P<ParameterName>$ to the script where P is the parameter indicator as shown below.

Before adding the Parameter to the script, create the desired parameter in **Tools > Parameters > Add Parameter Source** (context menu).




```
SET LOAD_FILE="$PSourcePath$\forecast.txt"
SET TRIG_FILE="$PSourcePath$\forecast.trg"
```

The script makes use of a number of environmental variables. These variable are acquired from both the Table and Connection properties. These variables are established in the environment by either WhereScape RED or the scheduler. If the script is to be executed outside of WhereScape RED or scheduler control, then these variables need to be assigned.

© Copyright, WhereScape Software Ltd All rights reserved.

The first section of the script defines the variables. The second section provides a timed wait for the load file to arrive. By default, the WAITSECS variable is set to zero so that no wait occurs. This can be set to a number of seconds that the script is to wait for the file to arrive.

```
REM ************************************************************
REM ************* W A I T     T I M E R ************************
REM ************************************************************
REM If a wait time specified then loop looking for the file
REM or trigger file to arrive until the wait time expires.
SET /A WAITSECS=0
:LABEL_WAITLOOP
if %WAITSECS% LEQ 0 goto LABEL_FILECHECK
REM see if the file exists and if so skip the wait check
if exist %LOAD_FILE% goto LABEL_FILECHECK
REM Call WslSleep to wait for 30 seconds. It will return remaining seconds
"C:\Program Files (x86)\WhereScape\Multi_Source_DW\WslSleep" %WAITSECS% 30
SET /A WAITSECS=%errorlevel%
goto LABEL_WAITLOOP
REM
REM Finished our wait loop. See if we have the file
REM ************************************************************
REM ************* F I L E    C H E C K ************************
REM ************************************************************
:LABEL_FILECHECK
if exist %LOAD_FILE% goto LABEL_TRIGGER
echo -1
echo File %LOAD_FILE% was not found.
exit
REM
```

Once the wait has completed, either through a time expiry or through the location of the file, we check that the file is present, and if not found report back a warning. This warning can be changed to an error by changing the first echo statement to "-2". See the **syntax section** for more information.

When a trigger file is specified the script looks for a trigger file, and will exit with the specified status if the file is not found. The following code is included if a trigger file is present.

```
REM
REM ************************************************************
REM ************* T R I G G E R ********************************
REM ************************************************************
:LABEL_TRIGGER
REM First clear any existing parameters
REM
isql -S %SERVER% -d %DATABASE% -U%USER% -P%PWD% -n -Q "UPDATE dss_parameter SET dss_parameter_value = NULL WHERE dss_parameter_name like 'FORECAST_%%'" -o %FILELOG%
REM
REM Get the first four and all remaining parameters
REM
for /F "tokens=1,2,3,4* delims=," %%a in (%TRIG_FILE%) do SET A=%%a&SET B=%%b&SET C=%%c&SET D=%%d&SET E=%%e
REM
REM For each parameter write it to the parameter table
REM
if defined A isql -S %SERVER% -d %DATABASE% -U%USER% -P%PWD% -n -Q "EXEC WsParameterWrite 'FORECAST_0','%A%','script load'"
if defined B isql -S %SERVER% -d %DATABASE% -U%USER% -P%PWD% -n -Q "EXEC WsParameterWrite 'FORECAST_1','%B%','script load'"
if defined C isql -S %SERVER% -d %DATABASE% -U%USER% -P%PWD% -n -Q "EXEC WsParameterWrite 'FORECAST_2','%C%','script load'"
if defined D isql -S %SERVER% -d %DATABASE% -U%USER% -P%PWD% -n -Q "EXEC WsParameterWrite 'FORECAST_3','%D%','script load'"
if defined E isql -S %SERVER% -d %DATABASE% -U%USER% -P%PWD% -n -Q "EXEC WsParameterWrite 'FORECAST_4','%E%','script load'"
REM
```

Such a file (trigger) contains control information about the main file to be loaded and arrives after the main file to indicate that the main file transfer has completed and that it is OK to load.

This section loads the contents of the trigger file into the Parameters table, so that the table can be validated. See the section on Post Load procedures for an explanation on how trigger files are used to validate a load file.

© Copyright, WhereScape Software Ltd All rights reserved.

```
REM ************************************************************
REM ************* L O A D    T H E   D A T A ******************
REM ************************************************************
echo load of %LOAD_TABLE% > %FILEAUD%
SET /A RESULT_CODE=1
SET RESULT_MSG=Load Completed
:LABEL_LOAD
SET FILE_NAME=NO_MORE_FILES
for %%A in (%LOAD_FILE%) do if "!FILE_NAME!"=="NO_MORE_FILES" SET FILE_NAME=%%A
if "!FILE_NAME!"=="NO_MORE_FILES" goto LABEL_EXIT
echo loading %FILE_NAME%  >> %FILEAUD%
echo --  > %FILECTL%
echo   BULK INSERT  >> %FILECTL%
if defined LOAD_SCHEMA (echo %LOAD_SCHEMA%%LOAD_TABLE% >> %FILECTL%) else echo %LOAD_TABLE% >> %FILECTL%
echo   FROM  >> %FILECTL%
echo     %FILE_NAME% >> %FILECTL%
echo   WITH >> %FILECTL%
echo   ( >> %FILECTL%
echo       TABLOCK >> %FILECTL%
echo       , DATAFILETYPE='char' >> %FILECTL%
echo     , CODEPAGE='raw' >> %FILECTL%
echo     , FIELDTERMINATOR = ',' >> %FILECTL%
echo     , FIRSTROW = 2 >> %FILECTL%
echo     , FORMATFILE='%FILEFMT%'>> %FILECTL%
echo   ) >> %FILECTL%
echo exit(SELECT @@ERROR) >> %FILECTL%
rem
rem In the following SQL statement use either -E for a trusted connection
rem or -Uxxx -Pyyy  where xxx is the username and ppp the password
rem
sqlcmd -S %SERVER% -d %LOAD_DB% -E -i %FILECTL% -o %FILELOG%
SET ERRLEV=%errorlevel%
IF %ERRLEV% EQU 0 GOTO LABEL_OKAY
:LABEL_FAIL
echo -2
echo Load failed with return status %ERRLEV% See error log for details
if exist %FILEAUD% type %FILEAUD% >&2
if exist %FILELOG% type %FILELOG% >&2
exit
:LABEL_OKAY
if exist %FILELOG% type %FILELOG% >> %FILEAUD%
rem
```

This section calls isql to invoke Bulk Insert to load the file. It makes use of a temporary file to build as a control file and then calls isql to load the data. Note that the load is actually in a for loop. Wild card file names can be used to load multiple files. Each file to be loaded must have the same format.

Note

The data being loaded is appended to the database table. As part of the scheduler run the load table is truncated if the property for truncation is set. In this way multiple files can be loaded into the database table.

Tip

If this script is to be executed outside the control of the WhereScape RED scheduler then a truncate statement may need to be performed on the database load table. This would normally be placed before the 'for loop' and would look something like the following:

isql -S %SERVER% -d %DATABASE% -U%USER% -P%PWD% -n -Q "truncate table load_forecast" -o %FILELOG%

This next section handles the rename and potential looping. The first block of code renames the file and also the trigger file if appropriate. This code is only generated if the rename fields in the file attributes are populated.

The goto label_load statement 9 lines from the end can be used if all the files in a wild card file load are required. Simply uncomment this goto statement and the script will load each file in the wild card.

© Copyright, WhereScape Software Ltd All rights reserved.

```
rem
REM ************************************************************
REM ************* R E N A M E ********************************
REM ************************************************************
:LABEL_RENAME
REM
REM rename the file
move %FILE_NAME% C:\loaded\
echo %FILENAME% moved to C:\loaded\ >> %FILEAUD%
REM rename the trigger file
move %TRIG_FILE% C:\loaded
echo %TRIG_FILE% moved to C:\loaded\ >> %FILEAUD%
REM
REM ************************************************************
REM ************* N E X T   F I L E ****************************
REM ************************************************************
REM If multiple files are being processed via a wildcard
REM uncomment the goto LABEL_LOAD statement
REM WARNING: Do not loop back unless the file name contains a wildcard
REM WARNING: You must rename the file that has been loaded if
REM          looping back otherwise the script will loop for
REM          ever loading the same file.
:LABEL_NEXTFILE
REM goto LABEL_LOAD
REM ************************************************************
REM ************* E X I T ************************************
REM ************************************************************
:LABEL_EXIT
echo %RESULT_CODE%
echo %RESULT_MSG%
if exist %FILEAUD% type %FILEAUD%
exit
```

Windows PowerShell Scripts

In addition to the conventional Windows scripting and other tools, WhereScape RED also supports Windows PowerShell scripts for loading data into a WhereScape RED managed Data Warehouse, as well as for exporting data from a WhereScape RED managed Data Warehouse.

The Windows PowerShell command line and scripting environment was introduced by Microsoft in Windows 7. For more information about PowerShell, please refer to the Microsoft TechNet website (https://technet.microsoft.com/en-us/library/dd742419.aspx).

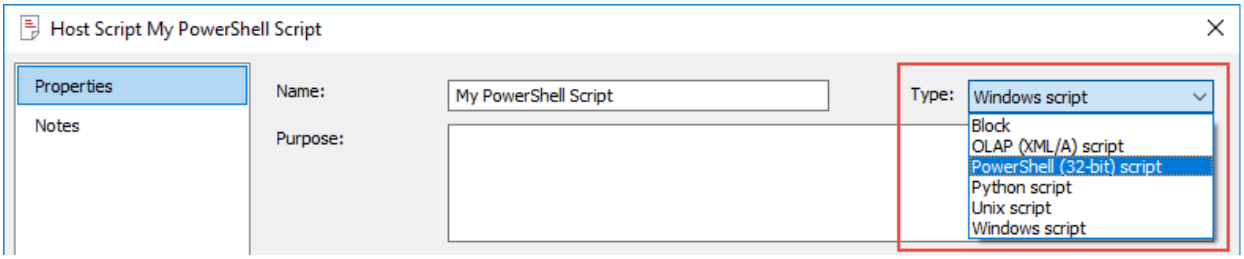
Note

WhereScape RED does not automatically create the PowerShell scripts, you must write them either directly via the WhereScape RED script editor or via a template that generates a PowerShell script. Each method is described below.

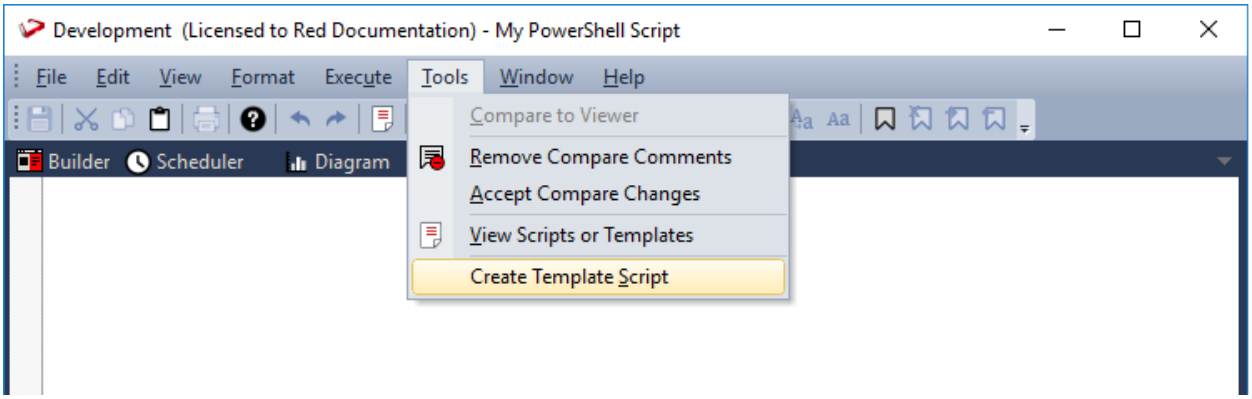
© Copyright, WhereScape Software Ltd All rights reserved.

Via the WhereScape RED Script Editor

You can manually write the PowerShell script from scratch, using the Script Editor. You need to create a Host Script object and set the script type to PowerShell script from the Host Script Properties screen.

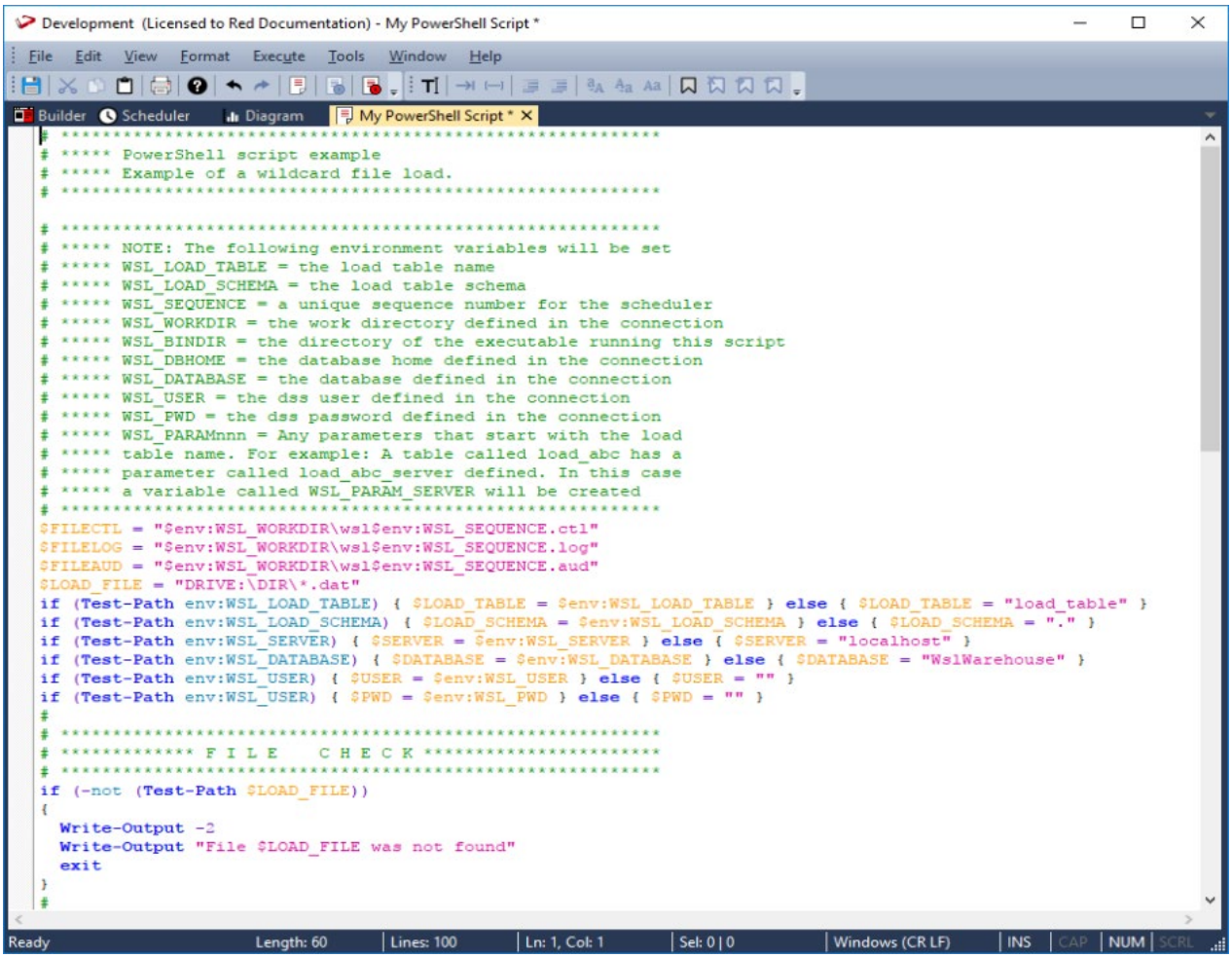


You can then open the blank Host Script created of type PowerShell Script to write your own script, or you can use the sample PowerShell scripts in WhereScape RED, which can be generated by selecting **Tools > Create Template Script** from the **Script Editor** screen.



The contents generated is an actual PowerShell script syntax that you can use to load data into a WhereScape RED managed Data Warehouse. You can also modify it and use it as a starting point to write your own PowerShell script for Load or Export objects.

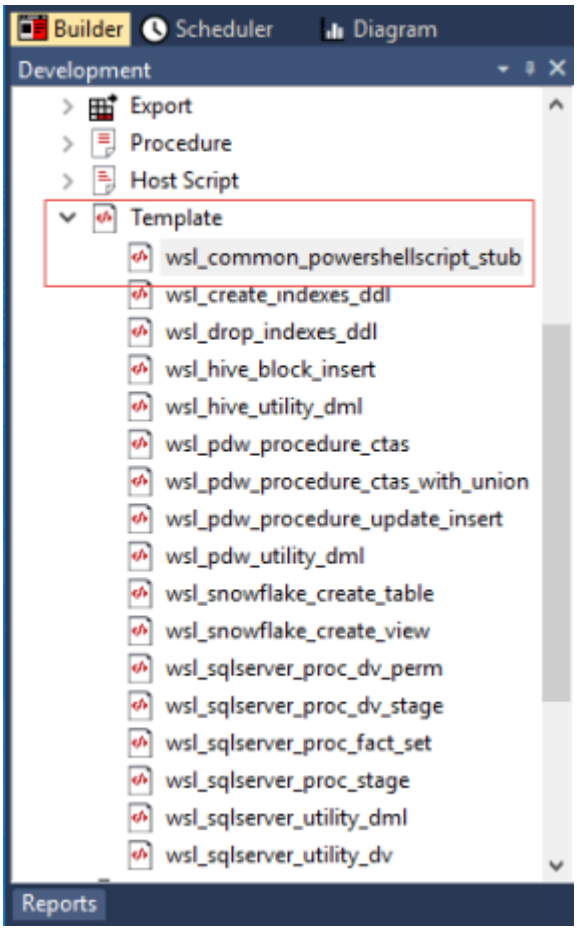
© Copyright, WhereScape Software Ltd All rights reserved.



© Copyright, WhereScape Software Ltd All rights reserved.

Via the Stub Template

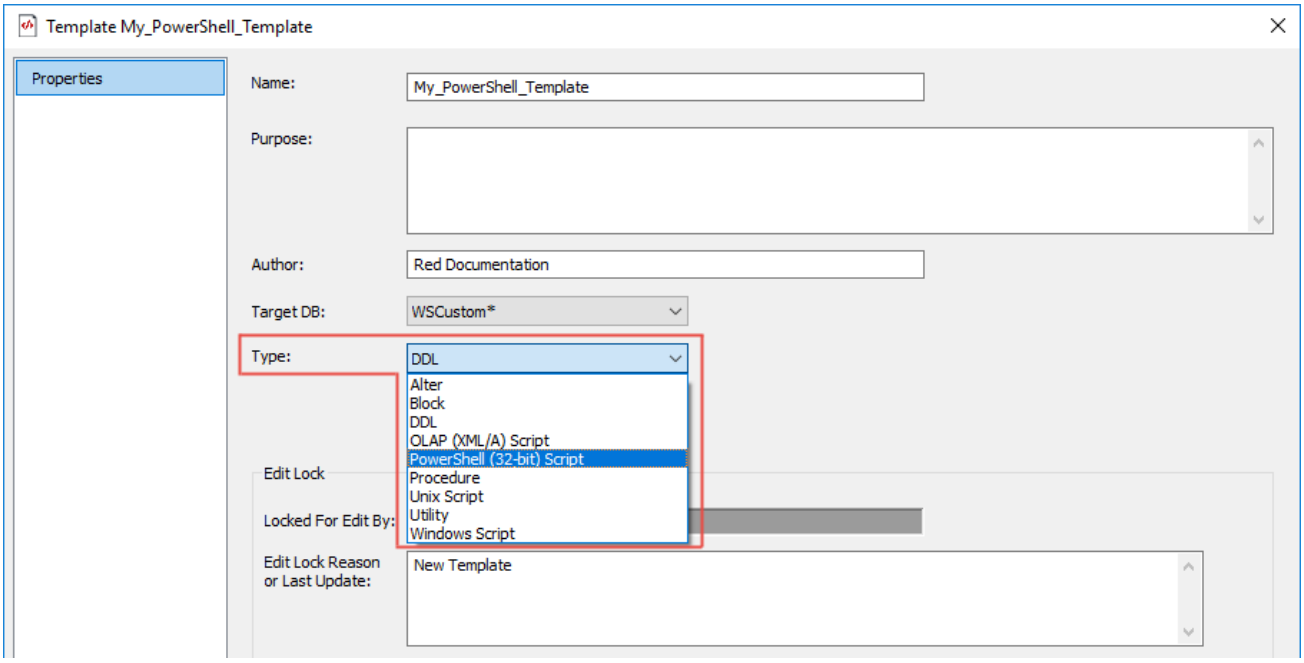
WhereScape RED also provides the Template object **wsl_common_powershellscript_stub** which is a basic PowerShell script type stub template that serves as guide on the use of a template to generate a PowerShell script.



Additional PowerShell Templates can be downloaded from the WhereScape website (https://www.wherescape.com/support/software-downloads-documentation/templates/).

© Copyright, WhereScape Software Ltd All rights reserved.

You can also create your own Template object and select PowerShell Script under the **Type** drop-down of the **Template Properties** screen.



Similar to the WhereScape RED Script Editor method described above, you can edit the template created to write your own script or open an existing PowerShell script to copy its contents to your template.

Note

WhereScape RED uses the 32-bit version of PowerShell by default. To use the 64-bit version, the following code must be added as the first line of the script. The script is relaunched and will be run in 64-bit version.

```
if ($pshome -like "*syswow64*") {
                & (join-path ($pshome -replace "syswow64", "sysnative") powershell.exe) -
file (join-path $psscriptroot $myinvocation.mycommand) @args
                exit
}
```

© Copyright, WhereScape Software Ltd All rights reserved.

Script Generation (UNIX/Oracle)

A sample UNIX script for Oracle is as follows. The key components of the script are described below:

```
#!/bin/sh
#*******************************************************************************
#****** Load of /home/dssadm/budget.txt into table load_budget_txt
#*******************************************************************************
#
LOAD_FILE=/home/dssadm/budget.txt
#
#*******************************************************************************
#************************** W A I T     T I M E R ***************************
#*******************************************************************************
# If a wait time specified then loop looking for the file or trigger
# to arrive until the wait period expires.
WAITSECS=0
if [ "$WAITSECS" -gt "0" ]
then
  # Get the current time and work out the end of the wait
  #
  NOW_HH=`date +%H`
  NOW_MM=`date +%M`
  NOW_SS=`date +%S`
  let NOW_HH="$NOW_HH * 3600"
  let NOW_MM="$NOW_MM * 60"
  let NOW_TIME="$NOW_HH + $NOW_MM + $NOW_SS"
  let TILL_TIME="$NOW_TIME + $WAITSECS"
  let START_TIME="$NOW_TIME"
  #
  # Loop checking for the file until the wait is complete
  #
  while [ "$NOW_TIME" -lt "$TILL_TIME" ]
  do
    FCOUNT=`find $LOAD_FILE -print 2>/dev/null | wc -l`
    if [ "$FCOUNT" -gt "0" ]
    then
      let NOW_TIME="$TILL_TIME"
    else
      #
      # if the file is not present sleep 30 seconds
      # and recalculate the time. If less than the start
      # then we have gone past midnight so add 86400
      sleep 30
```

© Copyright, WhereScape Software Ltd All rights reserved.

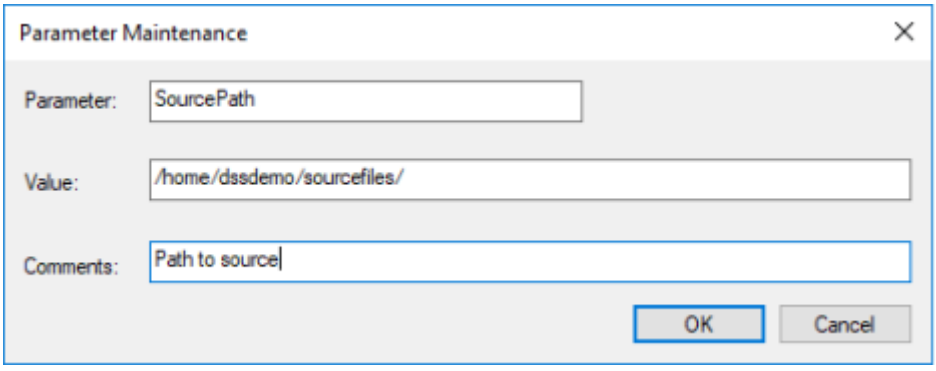
```
      NOW_HH=`date +%H`
      NOW_MM=`date +%M`
      NOW_SS=`date +%S`
      let NOW_HH="$NOW_HH * 3600"
      let NOW_MM="$NOW_MM * 60"
      let NOW_TIME="$NOW_HH + $NOW_MM + $NOW_SS"
      if [ "$NOW_TIME" -lt "$START_TIME" ]
      then
        let NOW_TIME="$NOW_TIME + 86400"
      fi
    fi
  done
fi
#
# Finished our wait loop. See if we have the file
#
#*************************************************************************
#************************* F I L E   C H E C K  ************************
#*************************************************************************
FCOUNT=`find $LOAD_FILE -print 2>/dev/null | wc -l`
if [ "$FCOUNT" -eq "0" ]
then
  #
  # file not present so exit with the requested return code
  #
  echo "-1"
  echo "File /home/dssadm/budget.txt was not found"
  exit
fi
```

Tip

Parameters can also be added to Scripts to facilitate deployment processes or environment changes without the need to regenerate scripts. Parameters can be added to scripts of Load and Export tables.

For example: add $P<ParameterName>$ to the script where P is the parameter indicator as show below.

Before adding the Parameter to the script, create the desired parameter in **Tools > Parameters> Add Parameter** (context menu option).




```
LOAD_FILE=$PSourcePath$/forecast.txt
TRIG_FILE=$PSourcePath$/forecast.trg
```

The script makes use of two environmental variables called DSS_USER and DSS_PWD. These variables are established in the environment by the scheduler. If the script is to be executed outside of scheduler control then these two variables will need to be assigned after the LOAD_FILE variable.

The first section of the script defines the load_file variable. The second section provides a timed wait for the load file to arrive. By default, the WAITSECS variable is set to zero so that no wait occurs. This can be set to a number of seconds that the script is to wait for the file to arrive.

© Copyright, WhereScape Software Ltd All rights reserved.

Once the wait has completed, either through a time expiry or through the location of the file, we check that the file is present, and if not found report back a warning. This warning can be changed to an error by changing the first echo statement to "-2". See the syntax section for more information.

When the file based load is used instead of the script based load, the scheduler looks for a trigger file if defined, and will exit with the specified status if the file is not found.

```
#*****************************************************************************
#************************ T R I G G E R  *************************************
#*****************************************************************************
#
# Clear any existing Trigger file parameters
#
### sqlplus -s <<EOF >/dev/null
### $DSS_USER/$DSS_PWD
### set sqlprompt "";
### set heading off;
### set pagesize 0
### set linesize 256
### set trimspool on
### set echo off;
### update dss_parameter set dss_parameter_value = NULL
### where dss_parameter_name like 'budget.txt_%';
### exit;
### EOF
### if [ "$?" -ne "0" ]
### then
###    echo "-1"
###    echo "Parameter clear Sqlplus returned a non standard return code of $?"
### fi
#
# Load the contents of the trigger file into parameters
#
### cat /home/dssadm/budget.trg | tr "\012" "\012" > /tmp/wsl2.trg
### TRIGNO=`cat /tmp/wsl2.trg | wc -l | tr -d " "`
### if [ "$TRIGNO" != "0" ]
### then
###   ROWNUM=0
###   while [ "$ROWNUM" -lt "$TRIGNO" ]
###   do
###     let ROWNUM="$ROWNUM+1"
###     TRIG_NO="budget.txt_$ROWNUM"
###     TRIG_MSG=`cat /tmp/wsl2.trg | head -$ROWNUM | tail -1 | sed "s/'/''/g"`
###     sqlplus -s <<EOF1 >/dev/null
###     $DSS_USER/$DSS_PWD
###     variable x number;
###     exec :x := WsParameterWrite('$TRIG_NO','$TRIG_MSG','load of /home/dssadm/budget.txt');
###     exit;
### EOF1
###   done
### fi
```

This section of the script is commented out by default. It is used when a trigger file is present. Such a file contains control information about the main file to be loaded and arrives after the main file to indicate that the main file transfer has completed and that it is OK to load.

This section loads the contents of the trigger file into the Parameters table, so that the table can be validated. See the section on Post Load procedures for an explanation on how trigger files are used to validate a load file.

© Copyright, WhereScape Software Ltd All rights reserved.

```
#*************************************************************************
#************************* L O A D   T H E   D A T A ********************
#*************************************************************************
#
# Loop through all files found loading them into the table
# Will need to use append in the sql load in case there are multiple files
#
FLIST=`find $LOAD_FILE -print`
for ACTFILE in $FLIST
do
  # Build the sql loader control script
  echo "load data" >/tmp/wsl2.ctl
  echo "infile '$ACTFILE'" >>/tmp/wsl2.ctl
  echo "badfile '/tmp/wsl2.bad'" >>/tmp/wsl2.ctl
  echo "append" >>/tmp/wsl2.ctl
  echo "into table load_budget_txt" >>/tmp/wsl2.ctl
  echo "fields terminated by \",\"" >>/tmp/wsl2.ctl
  echo "optionally enclosed by '\"'" >>/tmp/wsl2.ctl
  echo "trailing nullcols" >>/tmp/wsl2.ctl
  echo "(" >>/tmp/wsl2.ctl
  echo "order_number," >>/tmp/wsl2.ctl
  echo "order_line_number," >>/tmp/wsl2.ctl
  echo "product_code," >>/tmp/wsl2.ctl
  echo "customer_code," >>/tmp/wsl2.ctl
  echo "budget_quantity," >>/tmp/wsl2.ctl
  echo "budget_sales," >>/tmp/wsl2.ctl
  echo "budget_date date 'yyyy-mm-dd'" >>/tmp/wsl2.ctl
  echo ")" >>/tmp/wsl2.ctl
  ORACLE_SID=0817
  export ORACLE_SID
  sqlldr userid=$DSS_USER/$DSS_PWD control=/tmp/wsl2.ctl skip=1 silent=header,feedback log=/tmp/wsl2.log
  retcode=`echo $?`
  rows=`cat /tmp/wsl2.log| grep Rows | grep success`
```

This section calls sql*loader to load the file. It makes use of a temporary file to build as a control file and then calls sqlldr to load the data. Note that the load is actually in a for loop. Wild card file names can be used to load multiple files. Each file to be loaded must have the same format.

Note

The data being loaded is appended to the database table. As part of the scheduler run, the Load table is truncated if the property for truncation is set. In this way, multiple files can be loaded into the database table.

Tip

If this script is to be executed outside the control of the WhereScape RED scheduler, then a truncate statement may need to be performed on the database Load table. This would normally be placed before the 'for loop' and would look something like the following:

```
echo "truncate table load_budget_txt;" >/tmp/wsl2.sql
echo "commit;" >>/tmp/wsl2.sql
echo "exit" >>/tmp/wsl2.sql
sqlplus $DSS_USER/$DSS_PWD @/tmp/wsl2.sql >/dev/null
```

Note

The first echo has a single output '>' whereas the subsequent lines have two '>>'. The first output command creates or re-creates the file and the subsequent lines append to it. We put the output of the command to the null device as we do not want data in the output stream that does not match our syntax

© Copyright, WhereScape Software Ltd All rights reserved.

```
   case "$retcode" in
   0) echo "1";
      echo "load_budget_txt completed. $rows from $ACTFILE"
      ;;
   2) echo "-1";
      echo "Not all rows loaded. $rows. see error trail" ;
      cat /tmp/wsl2.log| grep Rows
      cat /tmp/wsl2.log>&2
      ;;
   1) echo "-2";
      echo "SQL*Loader execution exited with EX_FAIL, see error trail";
      cat /tmp/wsl2.log>&2
       ;;
   3) echo "-3";
      echo "SQL*Loader execution encountered a fatal error"
      cat /tmp/wsl2.log>&2
      ;;
   *) echo "-3";
      echo "SQL*Loader unknown return code"
      cat /tmp/wsl2.log>&2
      ;;
   esac
```

This section of the script checks the return code from the sql*loader command. Depending on the code it either identifies success, a warning or an error. In any non success situation it puts the log file out to the error stream '>&2'. This will result in the information ending up in the Detail/Error log when this script is executed through the scheduler.

```
   #
   #*************************************************************************
   #************************* R E N A M E ***********************************
   #*************************************************************************
   #
   # If we had a success or a warning then do a rename
   ### if [ "$retcode" -eq "0" -o "$retcode" -eq "2" ]
   ### then
     #
     # Rename the file
     #
   ###   mv $ACTFILE /home/dssadm/budget.txt
   ###   echo "$ACTFILE renamed to /home/dssadm/budget.txt"
     #
     # Rename the trigger
     #
   ###   mv /home/dssadm/budget.trg /home/dssadm/budget.trg
   ###   echo "/home/dssadm/budget.trg renamed to /home/dssadm/budget.trg"
   ### fi
 #*************************************************************************
 #************************* E N D    F I L E   L O A D   L O O P ********
 #*************************************************************************
 #
 # Uncomment the break statement if you only wish to process one file at
 # a time in a wildcard file load. The break will case the script to finish
 # after the first file that matches the wildcard has been loaded
 #
 # break
 done
 exit
```

This final section is largely commented out. It ends with the closing of the for loop that processes each file in a wild card file load.

The first block of commented out code renames the file and also the trigger file if appropriate. If this action is required, enter the rename path etc. and un-comment this code.

© Copyright, WhereScape Software Ltd All rights reserved.

The break statement 3 lines from the end can be used if just the first file in a wild card file load is required. Simply uncomment this break statement and the script will end after the first file has been loaded.

Script Generation (Windows/Oracle)

A sample Windows script for Oracle is as follows. The key components of the script are described below:

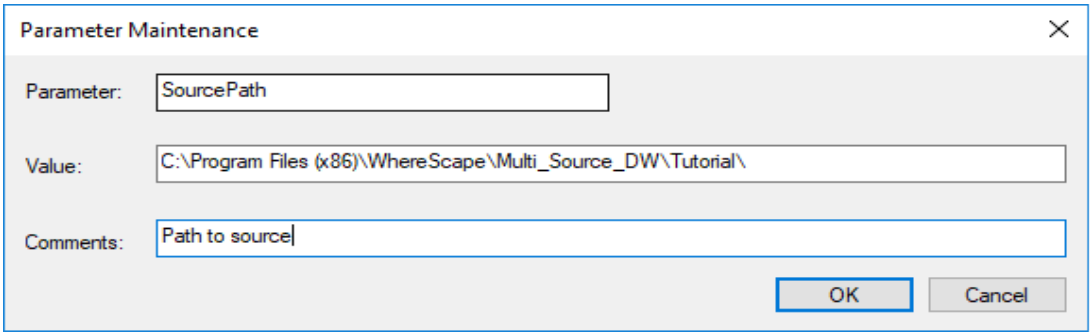
```
@echo off
setlocal enabledelayedexpansion
setlocal enableextensions
REM ************************************************************
REM ***** Load of D:\forecast.txt into load_forecast_txt
REM ************************************************************

REM ************************************************************
REM ***** NOTE: The following environment variables will be set
REM ***** WSL_LOAD_TABLE = the load table name
REM ***** WSL_SEQUENCE = a unique sequence number for the scheduler
REM ***** WSL_WORKDIR = the work directory defined in the connection
REM ***** WSL_DBHOME = the database home defined in the connection
REM ***** WSL_DATABASE = the database defined in the connection
REM ***** WSL_USER = the dss user defined in the connection
REM ***** WSL_PWD = the dss password defined in the connection
REM ***** WSL_PARAMnnn = Any parameters that start with the load
REM ***** table name. For example: A table called load_abc has a
REM ***** parameter called load_abc_server defined. In this case
REM ***** a variable called WSL_PARAM_SERVER will be created
REM ************************************************************

SET LOAD_FILE=D:\forecast.txt
SET TRIG_FILE=D:\forecast.trg

if defined WSL_LOAD_TABLE (SET LOAD_TABLE=%WSL_LOAD_TABLE%) else SET LOAD_TABLE=load_forecast_txt
if defined WSL_DBHOME (SET DBHOME=%WSL_DBHOME%) else SET DBHOME=
if defined WSL_DATABASE (SET DATABASE=%WSL_DATABASE%) else SET DATABASE=
if defined WSL_USER (SET USER=%WSL_USER%) else SET USER=dssadm
if defined WSL_USER (SET PWD=%WSL_PWD%) else SET PWD=wsl
if defined WSL_WORKDIR (SET WORKDIR=%WSL_WORKDIR%) else goto LABEL_FILE_DEFAULT
if defined WSL_SEQUENCE (SET SEQUENCE=%WSL_SEQUENCE%) else goto LABEL_FILE_DEFAULT
SET FILECMD=%WORKDIR%\wsl%SEQUENCE%.cmd
SET FILECTL=%WORKDIR%\wsl%SEQUENCE%.ctl
SET FILELOG=%WORKDIR%\wsl%SEQUENCE%.log
SET FILEAUD=%WORKDIR%\wsl%SEQUENCE%.aud
SET FILEBAD=%WORKDIR%\wsl%SEQUENCE%.bad
SET FILELO2=%WORKDIR%\wsl%SEQUENCE%.lo2
goto LABEL_WAIT
:LABEL_FILE_DEFAULT
SET FILECMD=c:\temp\wsl22.cmd
SET FILECTL=c:\temp\wsl22.ctl
SET FILELOG=c:\temp\wsl22.log
SET FILEAUD=c:\temp\wsl22.aud
SET FILEBAD=c:\temp\wsl22.bad
SET FILELO2=c:\temp\wsl22.lo2
:LABEL_WAIT
```

Tip

Parameters can also be added to Scripts to facilitate deployment processes or environment changes without the need to regenerate scripts. Parameters can be added to scripts of Load and Export tables.

For example: add $P<ParameterName>$ to the script where P is the parameter indicator as shown below.

Before adding the Parameter to the script, create the desired parameter in **Tools > Parameters > Add Parameter** (context menu option).




```
SET LOAD_FILE="$PSourcePath$\forecast.txt"
SET TRIG_FILE="$PSourcePath$\forecast.trg"
```

The script makes use of a number of environmental variables. These variables are acquired from both the table and connection properties. These variables are established in the environment by either WhereScape

© Copyright, WhereScape Software Ltd All rights reserved.

RED or the scheduler. If the script is to be executed outside of WhereScape RED or scheduler control then these variables will need to be assigned.

The first section of the script defines the variables. The second section provides a timed wait for the load file to arrive. By default the WAITSECS variable is set to zero, so that no wait occurs. This can be set to a number of seconds that the script is to wait for the file to arrive.

```
REM ************************************************************
REM ************** W A I T     T I M E R ***********************
REM ************************************************************
REM If a wait time specified then loop looking for the file
REM or trigger file to arrive until the wait time expires.
SET /A WAITSECS=0
:LABEL_WAITLOOP
if %WAITSECS% LEQ 0 goto LABEL_FILECHECK
REM see if the file exists and if so skip the wait check
if exist %TRIG_FILE% goto LABEL_FILECHECK
Call WslSleep to wait for 30 seconds. It will return remaining seconds
D:\Pms\Release\WslSleep %WAITSECS% 30
SET /A WAITSECS=%errorlevel%
goto LABEL_WAITLOOP
REM
REM Finished our wait loop. See if we have the file
REM ************************************************************
REM ************** F I L E    C H E C K ************************
REM ************************************************************
:LABEL_FILECHECK
if exist %TRIG_FILE% goto LABEL_TRIGGER
echo -1
echo File %TRIG_FILE% was not found.
exit
```

Once the wait has completed, either through a time expiry or through the location of the file, we check that the file is present, and if not found report back a warning. This warning can be changed to an error by changing the first echo statement to "-2". See the syntax section for more information.

When a trigger file is specified the script looks for a trigger file, and will exit with the specified status if the file is not found. The following code is included if a trigger file is present.

```
REM
REM ************************************************************
REM ************** T RI G G E R ********************************
REM ************************************************************
:LABEL_TRIGGER
REM
REM First clear any existing parameters
REM
echo UPDATE dss_parameter SET dss_parameter_value = NULL > %FILECMD%
echo WHERE dss_parameter_name like 'FORECAST_%%'; >> %FILECMD%
echo variable X varchar2 >> %FILECMD%
REM
REM Get the first four and all remaining parameters
REM
for /F "tokens=1,2,3,4* delims=," %%a in (%TRIG_FILE%) do SET A=%%a&SET B=%%b&SET C=%%c&SET D=%%d&SET E=%%e
REM
REM For each parameter write it to the parameter table
REM
if defined A echo EXEC :X := WsParameterWrite('FORECAST_0','%A%','script load'); >> %FILECMD%
if defined B echo EXEC :X := WsParameterWrite('FORECAST_1','%B%','script load'); >> %FILECMD%
if defined C echo EXEC :X := WsParameterWrite('FORECAST_2','%C%','script load'); >> %FILECMD%
if defined D echo EXEC :X := WsParameterWrite('FORECAST_3','%D%','script load'); >> %FILECMD%
if defined E echo EXEC :X := WsParameterWrite('FORECAST_4','%E%','script load'); >> %FILECMD%
REM
REM Execute the sql command
REM
echo exit; >> %FILECMD%
sqlplus -s %USER%/%PWD%@%DATABASE% @%FILECMD% > %FILELO2% 2>&1
REM
```

Such a file (trigger) contains control information about the main file to be loaded and arrives after the main file to indicate that the main file transfer has completed and that it is OK to load.

This section loads the contents of the trigger file into the Parameters table, so that the table can be validated. See the section on Post Load procedures for an explanation on how trigger files are used to validate a load file.

© Copyright, WhereScape Software Ltd All rights reserved.

```
REM ************************************************************
REM ************* L O A D    T H E   D A T A ******************
REM ************************************************************
echo load of %LOAD_TABLE% > %FILEAUD%
SET /A RESULT_CODE=1
SET RESULT_MSG=Load Completed
:LABEL_LOAD
SET FILE_NAME=NO_MORE_FILES
for %%A in (%LOAD_FILE%) do if "!FILE_NAME!"=="NO_MORE_FILES" SET FILE_NAME=%%A
if %FILE_NAME%==NO_MORE_FILES goto LABEL_EXIT
echo loading %FILE_NAME% >> %FILEAUD%
echo load data > %FILECTL%
echo infile '%FILE_NAME%' >> %FILECTL%
echo badfile '%FILEBAD%' >> %FILECTL%
echo into table %LOAD_TABLE% >> %FILECTL%
echo fields terminated by "," >> %FILECTL%
echo optionally enclosed by '^"' >> %FILECTL%
echo trailing nullcols >> %FILECTL%
echo ( >> %FILECTL%
echo  product_code,>> %FILECTL%
echo  customer_code,>> %FILECTL%
echo  forecast_quantity,>> %FILECTL%
echo  forecast_sales_value,>> %FILECTL%
echo  forecast_date date 'dd-mon-yyyy'>> %FILECTL%
echo ) >> %FILECTL%
sqlldr userid=%USER%/%PWD%@%DATABASE% control=%FILECTL% skip=1 silent=header,feedback log=%FILELOG%
SET /A ERRLEV=%errorlevel%
```

This section calls sql*loader (bulk insert for SQL Server) to load the file. It makes use of a temporary file to build as a control file and then calls sqlldr to load the data. Note that the load is actually in a for loop. Wild card file names can be used to load multiple files. Each file to be loaded must have the same format.

Note

The data being loaded is appended to the database table. As part of the scheduler run the load table is truncated if the property for truncation is set. In this way multiple files can be loaded into the database table.

Tip

If this script is to be executed outside the control of the WhereScape RED scheduler then a truncate statement may need to be performed on the database load table. This would normally be placed before the 'for loop' and for Oracle would look something like the following:

```
echo truncate table load_budget_txt; >/tmp/wsl2.sql
echo commit; >>/tmp/wsl2.sql
echo exit >>/tmp/wsl2.sql
sqlplus %USER%/%PWD%@%DATABASE% @/tmp/wsl2.sql > %FILELO2 2>&1
```

Note

The first echo has a single output '>' whereas the subsequent lines have two '>>'. The first output command creates or re-creates the file and the subsequent lines append to it. We put the output of the command to the null device as we do not want data in the output stream that does not match our syntax.

© Copyright, WhereScape Software Ltd All rights reserved.

This next section handles the return status from the load. It returns the appropriate status and messages to the scheduler.

```
if %ERRLEV% GTR 3 goto LABEL_BADRESULT
if %ERRLEV% LSS 0 goto LABEL_BADRESULT
if %ERRLEV% EQU 0 goto LABEL_OKAY
if %ERRLEV% EQU 1 goto LABEL_WARNING
if %ERRLEV% EQU 2 goto LABEL_ERROR
if %ERRLEV% EQU 3 goto LABEL_FATAL
:LABEL_BADRESULT
echo -3
echo SQL*Loader execution encountered an unexpected fatal error
type %FILELOG% >&2
exit
:LABEL_ERROR
echo -2
echo SQL*Loader execution exited with EX_FAIL, see error trail
type %FILELOG% >&2
exit
:LABEL_FATAL
echo -3
echo SQL*Loader execution encountered a fatal error
type %FILELOG% >&2
exit
:LABEL_WARNING
SET /A RESULT_CODE=-1
SET RESULT_MSG=WARNING: Not all rows loaded. See error trail
type %FILELOG% >>&2
:LABEL_OKAY
type %FILELOG% >>&2
REM
```

This final section is as follows:

```
REM ************************************************************
REM ************* R E N A M E **********************************
REM ************************************************************
:LABEL_RENAME
REM
REM rename the file
move %FILE_NAME% D:\old_data
echo %FILE_NAME% moved to D:\old_data >> %FILEAUD%
REM rename the trigger file
move %TRIG_FILE% D:\old_data
echo %TRIG_FILE% moved to D:\old_data >> %FILEAUD%
REM
REM ************************************************************
REM ************* N E X T   F I L E ****************************
REM ************************************************************
REM If multiple files are being processed via a wildcard
REM uncomment the goto LABEL_LOAD statement
REM WARNING: Do not loop back unless the file name contains a wildcard
REM WARNING: You must rename the file that has been loaded if
REM          looping back otherwise the script will loop for
REM          ever loading the same file.
:LABEL_NEXTFILE
REM goto LABEL_LOAD
REM ************************************************************
REM ************* E X I T **************************************
REM ************************************************************
:LABEL_EXIT
echo %RESULT_CODE%
echo %RESULT_MSG%
type %FILEAUD%
exit
```

The first block of code renames the file and also the trigger file if appropriate. This code is only generated if the rename fields in the file attributes are populated.

The goto label_load statement 9 lines from the end can be used if all the files in a wild card file load are required. Simply uncomment this goto statement and the script will load each file in the wild card.

© Copyright, WhereScape Software Ltd All rights reserved.

Script Editing

WhereScape RED includes a script editor which allows the maintenance of any host scripts within the data warehouse. The editor is invoked by double-clicking on a script name in the left pane or by right-clicking the script name and then selecting **Edit the Script** from the context menu.

Similar to the procedure editor, the script editor includes the following features:

- bash/DOS/PowerShell syntax highlighting
- improved find feature (repeated find, up and down, etc.)
- toggle the display of line numbering
- a status bar that includes line number, line length, selected text length, etc.
- a context sensitive toolbar
- tab to space conversion and vice versa
- uppercase, lowercase, titlecase text conversion command
- increase/decrease test indent command
- bookmark support

Indenting code

The tab character inserts four spaces into the text. A shift/tab removes four spaces.

Cut, Copy, Paste and Delete

The normal Windows cut, copy, paste and delete functions are available either through the toolbar or via the right mouse click context menu.

Indenting a block of text

A selected block of text can be indented by four spaces by depressing the tab character. Each tab will indent by a further four spaces. A shift/tab will remove four spaces from the front of each line in the selected block.

Viewing other scripts

During the editing process it is possible to pop up a window containing other scripts. This window allows cut and paste operations. In such a way it can be used as a work area or as a source of code. Select the **Tools > View Script or Template** menu option to bring up the viewer window.

A number of drop-down list boxes can be chosen from. Once an item is selected the viewer loads the code and moves to the right side of the edit window. The various options are:

- Version: A previously saved version of this script. The description and time are shown.
- Script: Some other script stored within the WhereScape RED metadata.
- Template: A template script as defined in the WhereScape RED metadata.

Once an item is chosen, the viewer appears on the right side of the edit window. The viewer or the main edit window can receive the focus by clicking the mouse within the appropriate window. Code can be cut from one window and pasted into the other. Any changes made in the viewer window cannot be saved.

Notes

Editing, deleting or compiling Locked for Edit or opened procedures/scripts

- Procedures or scripts cannot be deleted if they are Locked for Edit by any user, checked out by another user or if there is another object that has the same associated procedures or scripts.
- Saving or Compiling in the procedure or script edit window cannot be performed if the procedures or scripts become Locked for Edit by other users after the edit window has been opened.
- Procedures or scripts cannot be deleted or modified after the editor window has been opened, unless the Edit Lock has been released. Edit Locks can be released by any user in the Script or Procedure Properties screen.
- To prevent updates, deletes and modifications to certain procedures or scripts, it is best to use the Check Out functionality instead. For more information about this functionality, refer to **Object Check Outs and Check Ins** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

Script Testing

When a host script is scheduled, it is run in the scheduler environment. Therefore a UNIX scheduler must be available to run a UNIX script and only a Windows scheduler can run a Windows script.

It is possible to test a script interactively. For a Windows script, this is achieved by running the script on the current PC. For a UNIX script, WhereScape RED will attempt to start a telnet window to the UNIX host (as defined in the Connection set for the script) and run the script within this telnet window.

A script is invoked via the **Execute > Execute the Script** menu option. You can also right click the script from the **Host Script** objects list on the left pane and select **Execute Script** from the context menu.

The output from the script is shown in a pop-up dialog box.

UNIX Scripts

UNIX scripts have the additional execution option being the menu option **Execute > Execute and Retain**. This option creates a Telnet window to the UNIX host, execute the script, display the results and leave the Telnet window up. Additional commands or a review of what occurred can then be undertaken directly in the Telnet window. When completed the Telnet window should be closed to return to the script editor.

Note

A connection must be correctly set-up to support UNIX script testing. Refer to the section on UNIX connections in the RED Installation Guide for details.

Script Syntax

There are a number of conventions that must be followed if a host script is to be used by the WhereScape scheduler. These conventions are:

1. The first line of data in 'standard out' must contain the resultant status of the script. Valid values are '1' to indicate success, '-1' to indicate a warning condition occurred but the result is considered a success, '-2' to indicate a handled error occurred and subsequent dependent tasks should be held, -3 to indicate an unhandled Failure and that subsequent dependent tasks should be held.
2. The second line of data in 'standard out' must contain a resultant message of no more than 256 characters.
3. Any subsequent lines in 'standard out' are considered informational and are recorded in the audit trail. The normal practice is to place a minimum of information in the audit trail. All bulk information should be output to 'standard error'.
4. Any data output to 'standard error' is written to the error/detail log. Both the audit log and detail log can be viewed from the WhereScape RED tool under the Scheduler window.

Windows Example for Oracle:

In the following example, the first line '@echo off' prevents unwanted information from being reported to standard out. A Sql*loader script file is built up (echo statements). The sqlldr command is then executed with the 'silent-header,feedback' option once again to prevent unwanted output.

```
@echo off
echo load data >c:\temp\wsl1.ctl
echo infile 'C:\Temp\budget.txt' >>c:\temp\wsl1.ctl
echo badfile 'c:\temp\wsl1.bad' >>c:\temp\wsl1.ctl
echo into table load_budget_txt >>c:\temp\wsl1.ctl
echo fields terminated by "," >>c:\temp\wsl1.ctl
echo optionally enclosed by '^"' >>c:\temp\wsl1.ctl
echo trailing nullcols >>c:\temp\wsl1.ctl
echo ( >>c:\temp\wsl1.ctl
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
echo  product_code,>>c:\temp\wsl1.ctl
echo  customer_code,>>c:\temp\wsl1.ctl
echo  budget_quantity,>>c:\temp\wsl1.ctl
echo  budget_sales_value,>>c:\temp\wsl1.ctl
echo  budget_date date 'dd-mon-yyyy'>>c:\temp\wsl1.ctl
echo ) >>c:\temp\wsl1.ctl
sqlldr userid=dssadm/wsl@ORCL control=c:\temp\wsl1.ctl skip=1 silent=header,feedback
log=c:\temp\wsl1.log
goto answer%errorlevel%
:answer0
echo 1
echo Load completed successfully
type c:\temp\wsl1.log >&2
exit
:answer2
echo -1
echo Not all rows loaded.  See error trail
type c:\temp\wsl1.log >&2
exit
:answer1
echo -2
echo SQL*Loader execution exited with EX_FAIL, see error trail
type c:\temp\wsl1.log >&2
exit
:answer3
echo -3
echo SQL*Loader execution encountered a fatal error
type c:\temp\wsl1.log >&2
exit
```

Script Environment Variables

The following environment variables are available for all script loads and script exports, both Windows and UNIX/Linux.

All load scripts

The following variables are available in all load scripts:

Windows variable	UNIX/Linux variable	Description
WSL_LOAD_FULLNAME	LOAD_FULLNAME	The fully-qualified load table name.
WSL_LOAD_TABLE	LOAD_TABLE	The unqualified load table name.
WSL_LOAD_SCHEMA	LOAD_SCHEMA	The schema for the load table. **Notes:** • A trailing dot is appended for SQL Server or Oracle due to historical usage. • A trailing dot is not appended for any other database type due to historical usage. However, it is better not to assume the trailing dot is or isn't appended by using the variable like this, when it is not empty:

© Copyright, WhereScape Software Ltd All rights reserved.

Windows variable	UNIX/Linux variable	Description
		<table><tr><th>OS</th><th>If no trailing dot is wanted</th><th>If a trailing dot is wanted</th></tr><tr><td>Windows</td><td>!WSL_LOAD_SCHEMA:.=!</td><td>!WSL_LOAD_SCHEM</td></tr><tr><td>UNIX/ Linux</td><td>${LOAD_SCHEMA%.}</td><td>${LOAD_SCHEMA%.}.</td></tr></table>
WSL_LOAD_DB	LOAD_DB	The name of the database for the load table.
WSL_TEMP_DB	TEMP_DB	• **Teradata:** The name of the database for load temporary tables. • **PDW:** The name of the staging database for the load. **Others:** Not Used.
WSL_TGT_DSN	TGT_DSN	The ODBC data source name (DSN) for the load table's storage connection.
WSL_TGT_SERVER	TGT_SERVER	The server for the load table's storage connection.
WSL_TGT_DBPORT	TGT_DBPORT	The database port for the load table's storage connection.
WSL_TGT_DBID	TGT_DBID	The **Database ID** property of the load table's storage connection. **For Teradata** this is the Teradata Director Program ID (TDPID).**For Oracle** this is the Oracle SID or TNS Name.
WSL_TGT_USER	TGT_USER	The user id for the load table's storage connection.
WSL_TGT_PWD	TGT_PWD	The password for the load table's storage connection.

All load scripts from Database or ODBC connections

In **addition** to the variables in the previous table, the following variables are available in all load scripts from Database or ODBC connections:

Windows variable	UNIX/Linux variable	Description
WSL_SRC_DSN	SRC_DSN	The ODBC data source name (DSN) for the source connection.
WSL_SRC_SERVER	SRC_SERVER	The server for the source connection.
WSL_SRC_DBPORT	SRC_DBPORT	The database port for the source connection.
WSL_SRC_DBID	SRC_DBID	The **Database ID** property of the source connection. **For Teradata** this is the Teradata Director Program ID (TDPID).**For Oracle** this is the Oracle SID or TNS Name.
WSL_SRC_DB	SRC_DB	The name of the database for the source connection.

© Copyright, WhereScape Software Ltd All rights reserved.

<table>
<tr><th>Windows variable</th><th>UNIX/Linux variable</th><th>Description</th></tr>
<tr><td>WSL_SRC_SCHEMA</td><td>SRC_SCHEMA</td><td>The Source Schema property of the load.
Note
The property is fetched without modification, so there may or may not be a trailing dot depending on how it is configured.
However, it is better not to assume the trailing dot is or isn't appended by using the variable like this, when it is not empty:
<table><tr><th>OS</th><th>If no trailing dot is wanted</th><th>If a trailing dot is wanted</th></tr><tr><td>Windows</td><td>!WSL_SRC_SCHEMA:.=!</td><td>!WSL_SRC_SCHEMA:.=!.</td></tr><tr><td>UNIX/ Linux</td><td>${SRC_SCHEMA%.}</td><td>${SRC_SCHEMA%.}.</td></tr></table></td></tr>
<tr><td>WSL_SRC_USER</td><td>SRC_USER</td><td>The user id for the source connection.</td></tr>
<tr><td>WSL_SRC_PWD</td><td>SRC_PWD</td><td>The password for the source connection.</td></tr>
</table>

© Copyright, WhereScape Software Ltd All rights reserved.

All export scripts

The following variables are available in all export scripts:

<table>
<tr><th>Windows variable</th><th>UNIX/Linux variable</th><th>Description</th></tr>
<tr><td>WSL_EXP_NAME</td><td>EXP_NAME</td><td>The export object name.</td></tr>
<tr><td>WSL_EXP_FULLNAME</td><td>EXP_FULLNAME</td><td>The fully-qualified export table name.</td></tr>
<tr><td>WSL_EXP_TABLE</td><td>EXP_TABLE</td><td>The unqualified export table name.
Note
For Windows script exports from PDW, this variable is initialized with the fully-qualified export table name, due to historical usage.
To enable this variable to be uniformly described and used as the unqualified export table name, an additional variable <code>WSL_EXP_SIMPLENAME</code> is created.This allows the following command to be explicitly added to the top of the script by the script author, to be executed before all other processing:
<code>if defined WSL_EXP_SIMPLENAME (SET WSL_EXP_TABLE=!WSL_EXP_SIMPLENAME!) else SET WSL_EXP_TABLE=</code>
After such a command is executed, the variable <code>WSL_EXP_TABLE</code> will contain the unqualified export table name.</td></tr>
<tr><td>WSL_EXP_SCHEMA</td><td>EXP_SCHEMA</td><td>The schema for the export table.
Note
A trailing dot is appended for SQL Server or Oracle due to historical usage. A trailing dot is not appended for any other database type due to historical usage.
However, it is better not to assume the trailing dot is or isn't appended by using the variable like this, when it is not empty:
<table><tr><th>OS</th><th>If no trailing dot is wanted</th><th>If a trailing dot is wanted</th></tr><tr><td>Windows</td><td>!WSL_EXP_SCHEMA:.=!</td><td>!WSL_EXP_SCHEMA:.=!.</td></tr><tr><td>UNIX/ Linux</td><td>${EXP_SCHEMA%.}</td><td>${EXP_SCHEMA%.}.</td></tr></table></td></tr>
<tr><td>WSL_EXP_DB</td><td>EXP_DB</td><td>The name of the database for the export table.</td></tr>
<tr><td>WSL_TEMP_DB</td><td>TEMP_DB</td><td>Teradata: The name of the database for export temporary tables.
Others: Not used.</td></tr>
<tr><td>WSL_SRC_DSN</td><td>SRC_DSN</td><td>The ODBC data source name (DSN) for the export table's storage connection.</td></tr>
<tr><td>WSL_SRC_SERVER</td><td>SRC_SERVER</td><td>The server for the export table's storage connection.</td></tr>
<tr><td>WSL_SRC_DBPORT</td><td>SRC_DBPORT</td><td>The database port for the export table's storage connection.</td></tr>
</table>

© Copyright, WhereScape Software Ltd All rights reserved.

Windows variable	UNIX/Linux variable	Description
WSL_SRC_DBID	SRC_DBID	• The **Database ID** property of the export table's storage connection. • **For Teradata** this is the Teradata Director Program ID (TDPID).**For Oracle** this is the Oracle SID or TNS Name.
WSL_SRC_USER	SRC_USER	The user id for the export table's storage connection.
WSL_SRC_PWD	SRC_PWD	The password for the export table's storage connection.

All scripts

In **addition** to the specific variables in the previous tables, the following variables are available in all scripts:

Windows variable	UNIX/Linux variable	Description
WSL_META_DSN	META_DSN	The ODBC data source name (DSN) for the meta-repository connection.
WSL_META_SERVER	META_SERVER	The server for the meta-repository connection.
WSL_META_DBID	META_DBID	The **Database ID** property of the meta-repository connection. **For Teradata** this is the Teradata Director Program ID (TDPID).**For Oracle** this is the Oracle SID or TNS Name.
WSL_META_DB	META_DB	The name of the database for the meta-repository connection.
WSL_META_SCHEMA	META_SCHEMA	The meta-repository table qualifier, with a trailing dot. • **For SQL Server and DB2** this is the schema for the meta-repository. • **For Teradata and Oracle** this is not actually a schema name. **Note** A trailing dot is appended due to historical usage. However, it is better not to assume the trailing dot is or isn't appended by using the variable like this, when it is not empty:

OS	If no trailing dot is wanted	If a trailing dot is wanted
Windows	!WSL_META_SCHEMA:.=!	!WSL_META_SCHEMA:.=!.
UNIX/ Linux	${META_SCHEMA%.}	${META_SCHEMA%.}.

WSL_META_USER	META_USER	The user id for the meta-repository connection.
WSL_META_PWD	META_PWD	The password for the meta-repository connection.
WSL_WORKDIR	WORKDIR	• **Windows:** The work directory defined in the Windows connection. • **UNIX/Linux:** The work directory defined in the UNIX/Linux or Hadoop connection.

© Copyright, WhereScape Software Ltd All rights reserved.

Windows variable	UNIX/Linux variable	Description
WSL_SEQUENCE	SEQ	A unique sequence number for the load or export task.
WSL_PARAMnnn	PARAMnnn	Any parameters that start with the load table or export object name. **Example:** A table called **load_abc** has a parameter called **load_abc_server** defined. In this case, a variable called **WSL_PARAM_SERVER** (Windows) or **PARAM_SERVER** (UNIX/Linux) will be created.

Calling a Batch File from a Script

Below is an example RED host script which calls a batch file:

```
@ECHO OFF
SETLOCAL ENABLEDELAYEDEXPANSION
SETLOCAL ENABLEEXTENSIONS
CALL c:\temp\MyBatchFile.bat > c:\temp\MyBatchFile.log 2>&1
IF %ERRORLEVEL% EQU 0 GOTO LABEL_OKAY
ECHO -2
ECHO Batch file returned an error code of %ERRORLEVEL%
TYPE c:\temp\MyBatchFile.log
EXIT
:LABEL_OKAY
ECHO 1
ECHO Batch file completed successfully
TYPE c:\temp\MyBatchFile.log
```

Where "`c:\temp\MyBatchFile.bat`" contains this:

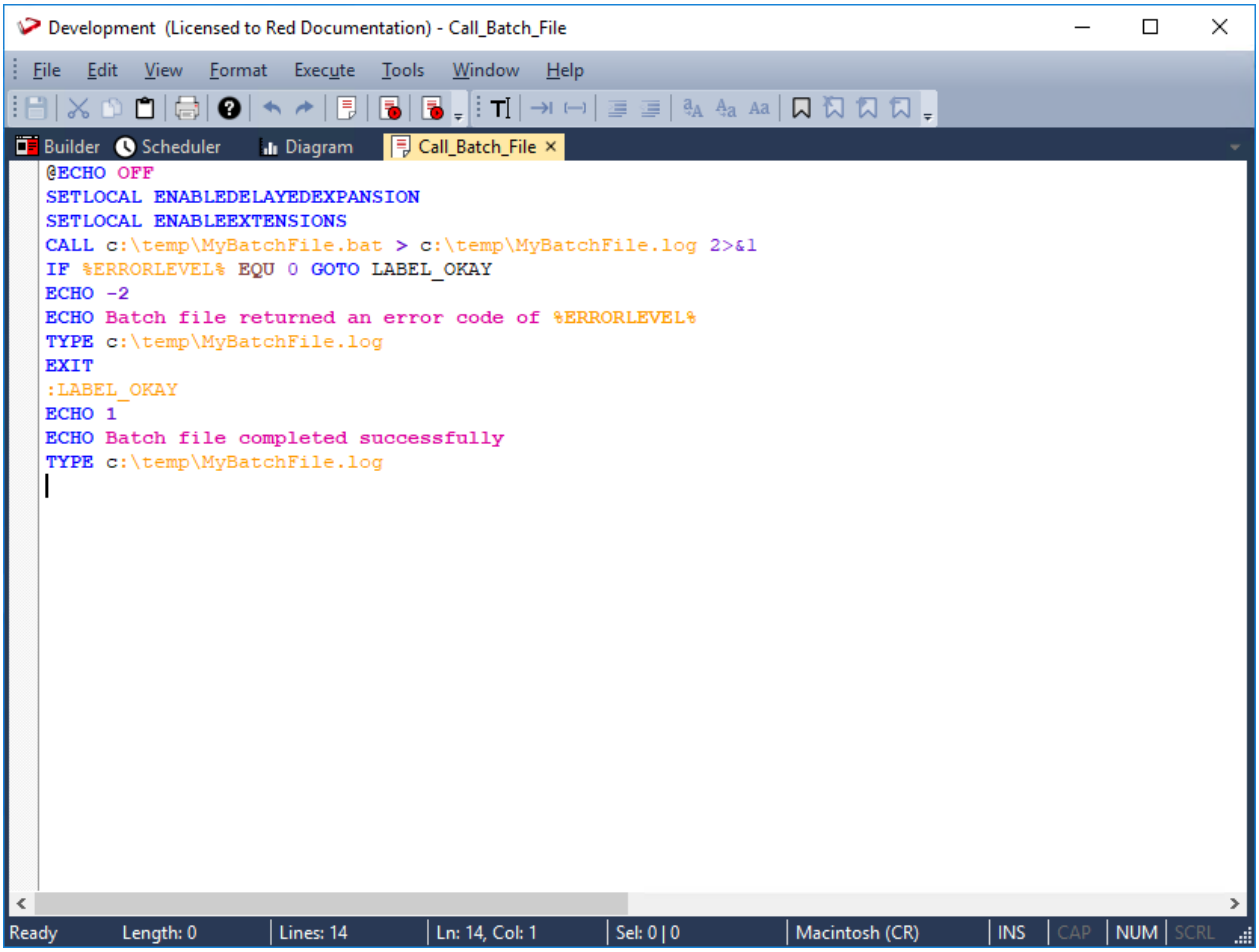
```
ECHO Hello
SET ERRORLEVEL=0
```

© Copyright, WhereScape Software Ltd All rights reserved.

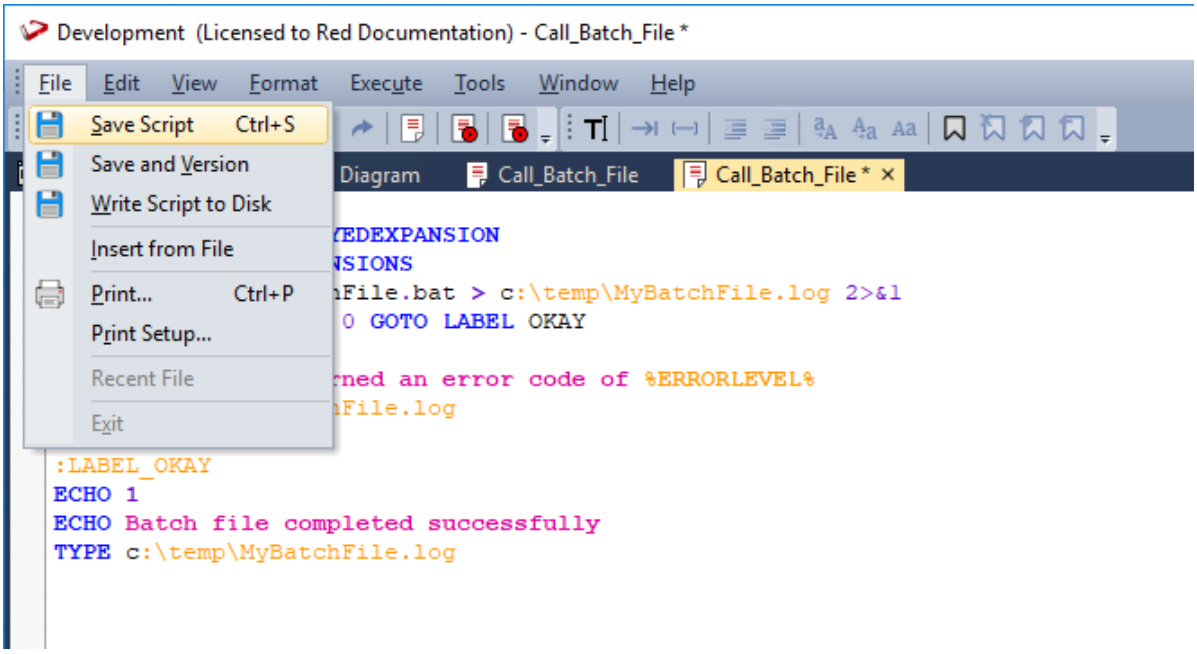
Create the Host Script in RED:

© Copyright, WhereScape Software Ltd All rights reserved.

Edit the Script and enter the following:

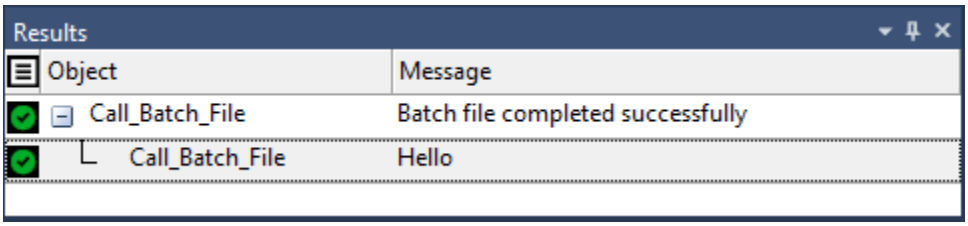


Save the Script:



© Copyright, WhereScape Software Ltd All rights reserved.

When the script is executed, you will see the following results:

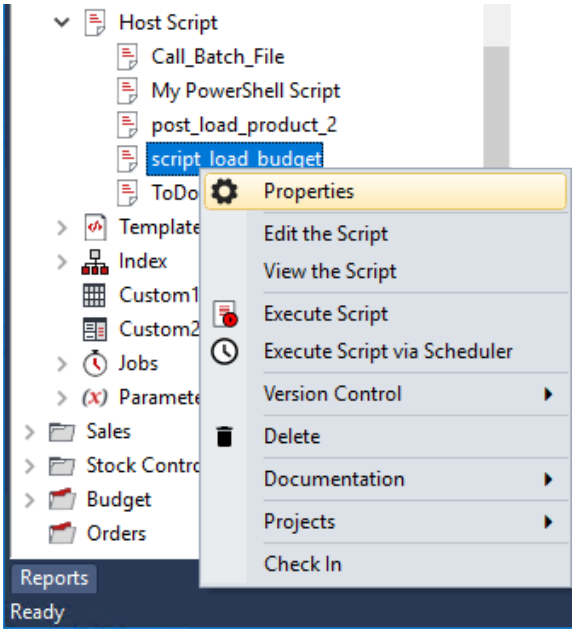


Scheduling Scripts

When a host script is scheduled, it is run in the scheduler environment. Therefore, a UNIX scheduler must be available to run a UNIX script and only a Windows scheduler can run a Windows script.

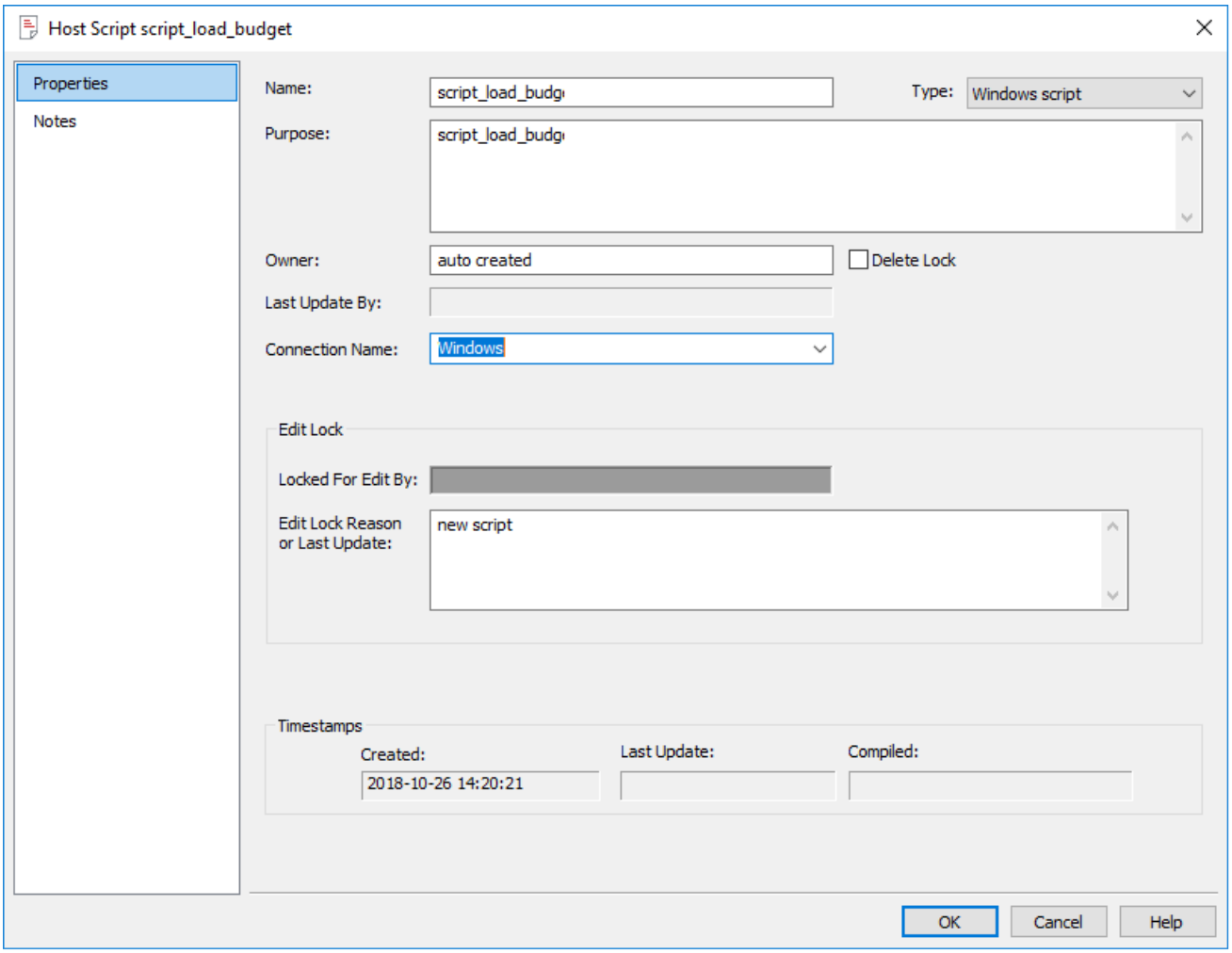
It is important to set the **connection** on the **Properties** screen for the script.

Right-click on the host script in the left pane and select **Properties**.



© Copyright, WhereScape Software Ltd All rights reserved.

Set the **Connection Name** to either **Windows** or **Unix** and then click **OK**.



Note

If you fail to set the default connection for the host script, you will receive a return message of **Invalid Host Type** when the host script is executed.

There are a number of conventions that must be followed if a host script is to be used by the WhereScape RED scheduler. These conventions are:

1. The first line of data in 'standard out' must contain the resultant status of the script. Valid values are '1' to indicate success, '-1' to indicate a warning condition occurred but the result is considered a success, '-2' to indicate a handled error occurred and subsequent dependent tasks should be held, -3 to indicate an unhandled Failure and that subsequent dependent tasks should be held.
2. The second line of data in 'standard out' must contain a resultant message of no more than 256 characters.
3. Any subsequent lines in 'standard out' are considered informational and are recorded in the audit trail. The normal practice is to place a minimum of information in the audit trail. All bulk information should be output to 'standard error'.
4. Any data output to 'standard error' is written to the error/detail log. Both the audit log and detail log can be viewed from the scheduler window.

© Copyright, WhereScape Software Ltd All rights reserved.

Manually Created Scripts

Individual scripts can also be manually created in RED to perform and schedule tasks that are not related to load tables.

The example below shows a minimal script that will run successfully.

```
@echo off
REM ************************************************************
REM **** Parameter Example
REM ************************************************************

SET /A RESULT_CODE=1
SET RESULT_MESSAGE="Success."

SET FILEAUD=%WORKDIR%\wsl%SCRIPT%%SEQUENCE%.aud
SET FILE_PROCESS_LOG=c:\temp\process.log

echo $PDS_Customer_Process_Date$ >> %FILE_PROCESS_LOG%

echo %RESULT_CODE%
echo %RESULT_MSG%

exit
```

Please note that you need to use the following codes to determine the script's results meaning. It is important that one of these codes is the first output of the script.

Output	Description
Result Number	Output Result Number: • 1 Success. • **-1 Warning.** • -2 Error. • -3 Fatal/Unexpected Error.

© Copyright, WhereScape Software Ltd All rights reserved.

Templates

Templates provide the ability to customize automatically generated code within RED. This feature is most suited to users that would like to customize automatically generated code or would like to expand RED to support non-native database platforms.

Creating templates is an advanced function that requires intimate knowledge of RED operations and metadata structure. WhereScape recommends that you contact our consulting team to assist with this feature. However, should you wish to use this feature independently, example templates and up-to-date reference information is available on our website:

https://www.wherescape.com/support/software-downloads-documentation/wherescape-red/templates/

Some templates may be included in your RED installation, depending on your license.

Each template is assigned a type and a target database, these properties are used to assist with filtering when associating table operations to templates. RED supports templates for the following operations:

Operation	Database	Template Type
Alter Table DDL	Custom (on SQL Server, Oracle and Teradata)	DDL
Create Table DDL	All database types	DDL
Create Index DDL	Custom (on SQL Server and Oracle)	DDL
Drop Index DDL	Custom (on SQL Server and Oracle)	DDL
Export Script	All database types	Windows Script
		PowerShell Script
		Unix Script
		OLAP/XMLA Script
Load Script	All database types	Windows Script
		PowerShell Script
		Unix Script
		OLAP/XMLA Script
Update Procedure	Custom (on SQL Server, Oracle and Teradata)	Block
		PowerShell Script
	Hive	Block
	SQL Server, Teradata, Oracle and PDW	Block
		Procedure
Custom Update Procedure	SQL Server, Teradata and Oracle	Procedure
Update Script	Custom (on SQL Server, Oracle and Teradata)	PowerShell Script
Post Load Procedure	Custom (on SQL Server, Oracle and Teradata)	Block
	Hive	Block
	SQL Server, Teradata, Oracle and PDW	Block
	SQL Server, Teradata and Oracle	Procedure
Post Load Script	Custom (on SQL Server, Oracle and Teradata)	PowerShell Script

Notes:

- Script-based loads and exports on Windows, supports both DOS Batch and **PowerShell scripts**.
- Refer to **Rebuilding Update Procedures** for details on generating an update procedure via a template.

© Copyright, WhereScape Software Ltd All rights reserved.

Utility type templates contain common code for use by other templates.

Templates are written in the Pebble template language—refer to **Pebble Templates** and **Pebble Syntax** sections for details.

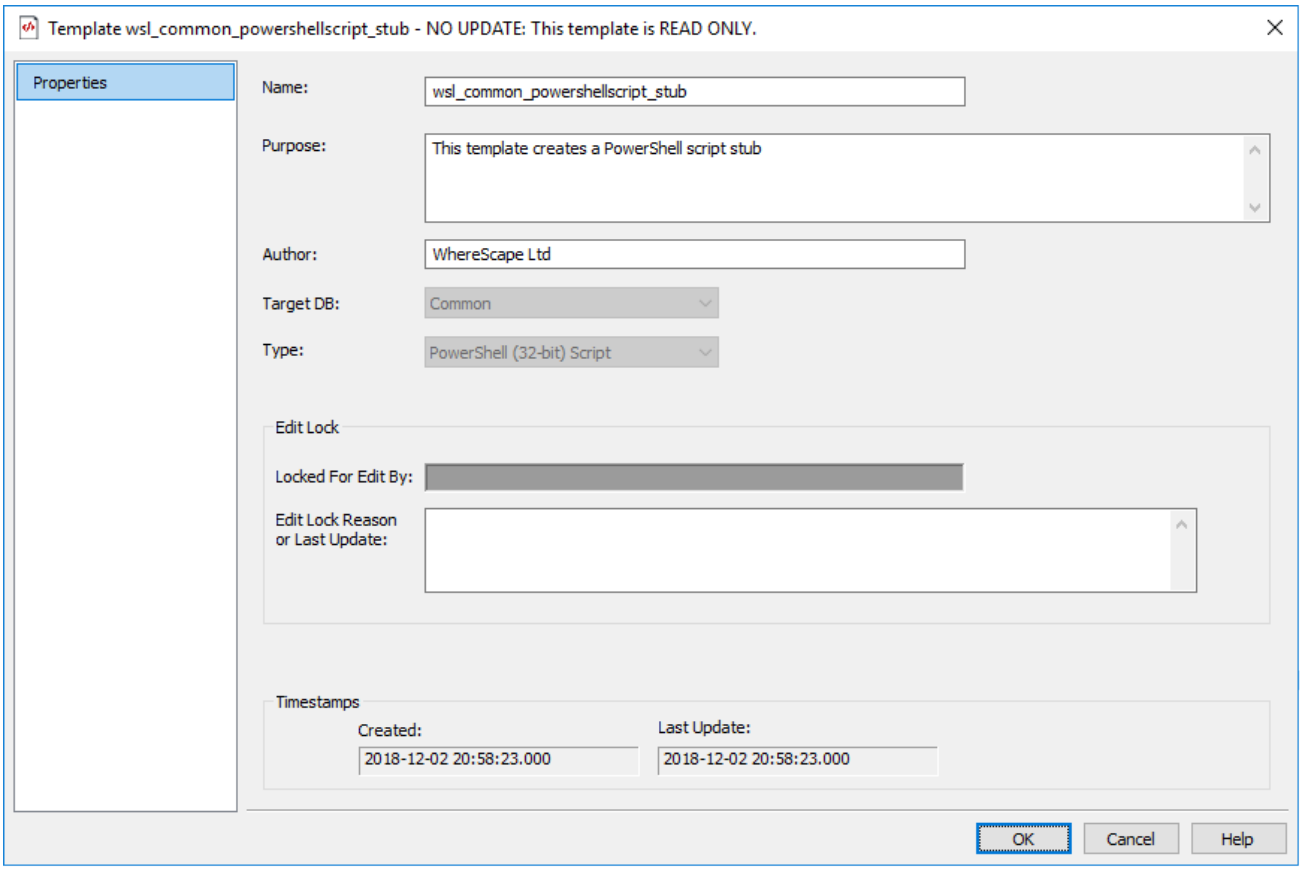
You can also define default templates to use for the update routines of table objects in RED by Connection type—refer to **Connection Routine Templates** for details.

Tip:

Detailed logs can be produced during template evaluation by typing **FULLLOG** in the Notes of the relevant connection.

Template Properties

The properties screen for a template is shown below.



Name, **Purpose** and **Author** fields should be completed to provide background information on the template; these fields are purely informational.

Created and **Last Update** fields provide date information on the template.

The **Target DB** sets the type of database connections for this template. The template is only selectable for an operation if the **Target DB** field matches that of the object. Target DB is restricted, based on your license.

- Common (applies to all databases)
- DB2
- Greenplum
- Hive
- Netezza
- Oracle
- PDW
- SQL Server
- Tabular
- Teradata
- Custom

© Copyright, WhereScape Software Ltd All rights reserved.

Note

Hive and Custom update procedure templates only support Block update procedures so you should create a block template for these.

The **Type** field specifies what the template can be used for in RED. This can be set to one of the following:

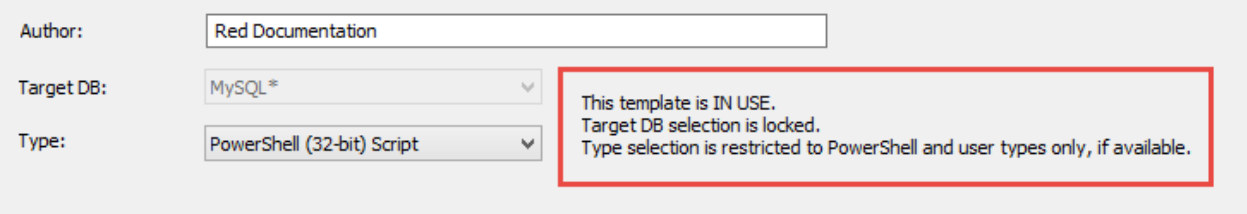
- Alter
- Block
- DDL
- Function
- OLAP (XML/A) Script
- PowerShell Script
- Procedure
- Unix Script
- Utility
- Windows Script

Refer to **Template Usage** for more details.

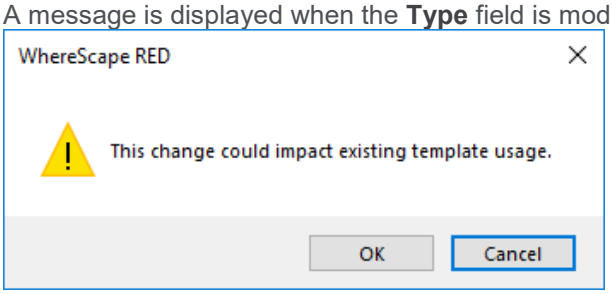
Note

Additional host script language types which can be applied to templates can be defined in RED via the **Tools > Host Script Languages** option.

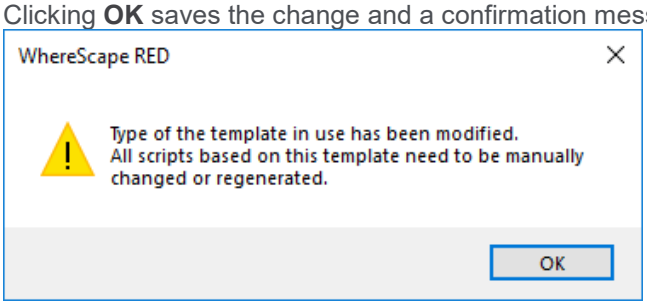
The **Type** field setting of a template that is currently IN USE can be changed but is restricted to PowerShell and user defined script types only.



A message is displayed when the **Type** field is modified:



Clicking **OK** saves the change and a confirmation message is displayed:



The **Edit Lock** fields provide information if the template is currently locked for editing by another user.

Templates cannot be deleted or modified after the editor window has been opened, unless the Edit Lock has been released. Edit Locks can be released by any user in the template Properties screen.

© Copyright, WhereScape Software Ltd All rights reserved.

Properties Screen Banner

The banner message varies depending on the state of the template object.

- *<TEMPLATE NAME> - NO UPDATE: This template is READ ONLY*
  indicates that this is a shipped master template that may not be changed.
- *<TEMPLATE NAME> - NO UPDATE: Checked out by <USER>*
  indicates that the template has been checked out by a user and thus may not be edited until checked back in.
- *<TEMPLATE NAME> - NO UPDATE: Checked out by <USER>*
  indicates that the template is in use by procedures and therefore cannot be altered. The **Type** field is also locked. See notes above for exceptions.

Pebble Templates

Pebble is a scripting language that generates text output, based on provided meta data. The usage of Pebble templates in WhereScape RED does not require knowledge of Java. When looking at the Pebble documentation ignore the references to Java, including information about setting up Pebble in Java.

Operation Types

Create DDL

- env
- table
- options
- settings
- Types

Custom Procedure

- env
- table
- options
- settings
- Types

Export Script

- env
- table
- options
- settings
- Types

Index Create DDL

- env
- index
- options
- settings
- Types

Index Drop DDL

- env
- index
- options
- settings
- Types

© Copyright, WhereScape Software Ltd All rights reserved.

Load Script

- env
- table
- options
- settings
- Types

Table Alter DDL

- env
- changes
- actualTable
- table
- options
- settings
- Types

Update Procedure

- env
- table
- options
- settings
- Types

Types

ActualTable

Name	Nullable	Type	Notes
catalog		Scalar	
schema		Scalar	
name		Scalar	
additionalInformation		Array of **AdditionalDatum**	
columns		Array of **Column**	

AdditionalDatum

Name	Nullable	Type	Notes
key		Scalar	
description		Scalar	
value		Scalar	

ChangeType

Name	Nullable	Type	Notes
Types.ChangeType.TableCreation		**TableCreation**	
Types.ChangeType.TableAdditionalInformationChange		**TableAdditionalInformationChange**	
Types.ChangeType.ColumnAddition		**ColumnAddition**	

© Copyright, WhereScape Software Ltd All rights reserved.

Name	Nullable	Type	Notes
Types.ChangeType.ColumnDeletion		ColumnDeletion	
Types.ChangeType.ColumnNameChange		ColumnNameChange	
Types.ChangeType.ColumnDataTypeChange		ColumnDataTypeChange	
Types.ChangeType.ColumnAdditionalInformationChange		ColumnAdditionalInformationChange	

Column

Name	Nullable	Type	Notes
additive		Scalar	
artificial		Scalar	
baseDataType		Scalar	
businessKey		Scalar	
changeHashKey		Scalar	
columnId		Scalar	
createTime		Scalar	
currentFlag		Scalar	
dataType		Scalar	
dataTypeScale		Scalar	
dataTypeSize		Scalar	
defaultValue		Scalar	
description		Scalar	
displayName		Scalar	
dss		Scalar	
dssEndDate		Scalar	
dssLoadDate		Scalar	
dssRecordSource		Scalar	
dssStartDate		Scalar	
dssVersion		Scalar	
format		Scalar	
fullDataType		Scalar	
hashKeyImmediateSources	Y	Array of column	
hashKeySources	Y	Array of column	
hubHashKey		Scalar	
keyType		enum	
linkHashKey		Scalar	
lookupKey		Scalar	
name		Scalar	
nullAllowed		Scalar	
numeric		Scalar	
partition		Scalar	
rangeEnd		Scalar	

© Copyright, WhereScape Software Ltd All rights reserved.

Name	Nullable	Type	Notes
rangeStart		Scalar	
slowlyChanging		Scalar	
source		Scalar	
sourceColumn	Y	Link to column identified by name	
sourceSystem		Scalar	
sourceTable	Y	Link to table identified by name	
table		table	
transform		Scalar	
transformType		enum	
updateTime		Scalar	
zeroKeyValue		Scalar	

ColumnAddition

Name	Nullable	Type	Notes
columnName		Scalar	

ColumnAdditionalInformationChange

Name	Nullable	Type	Notes
columnName		Scalar	
changedKey		Scalar	
changedDescription		Scalar	
expectedValue		Scalar	
actualValue		Scalar	

ColumnDataTypeChange

Name	Nullable	Type	Notes
columnName		Scalar	
dataType		Scalar	
actualDataType		Scalar	

ColumnDeletion

Name	Nullable	Type	Notes
columnName		Scalar	

ColumnJoin

Name	Nullable	Type	Notes
fromColumn		Link to column identified by name	
toColumn		Link to column identified by name	

© Copyright, WhereScape Software Ltd All rights reserved.

ColumnNameChange

Name	Nullable	Type	Notes
columnName		Scalar	
actualColumnName		Scalar	

DateBasedPartitioningInfo

Name	Nullable	Type	Notes
partitionType		enum	
dateDimensionColumn	Y	Link to column identified by name	
startDateKey		Scalar	
endDateKey		Scalar	

EnumValue

Name	Nullable	Type	Notes
name		Scalar	
code		Scalar	

Env

Name	Nullable	Type	Notes
licensedTo		Scalar	
productVersion		Scalar	
userName		Scalar	
metaSchemaPrefix		Scalar	
loginName		Scalar	
currentTimestamp		Scalar	

ExportInfo

Name	Nullable	Type	Notes
connectionType		enum	
exportConnection		Link to connection identified by name	
sourceDbType		enum	
file		**FileInfo**	
routine		Scalar	
format		Scalar	
headerRow	Y	enum	
options		Scalar	
triggerFile		**FileInfo**	
triggerParameter1		Scalar	
triggerParameter2		Scalar	

© Copyright, WhereScape Software Ltd All rights reserved.

Name	Nullable	Type	Notes
triggerParameter3		Scalar	
compressAfterExport		Scalar	
compressUtility		**FileInfo**	
compressParameters		Scalar	
whereClause		Scalar	

ExtendedPropertyDefinition

Name	Nullable	Type	Notes
id		Scalar	
variableName		Scalar	
displayName		Scalar	
description		Scalar	
isMasked		Scalar	
scopes		Array of **ExtendedPropertyScope**	

ExtendedPropertyScope

Name	Nullable	Type	Notes
id		Scalar	
connectionType		enum	
objectType		enum	
databaseType		enum	

FK

Name	Nullable	Type	Notes
fromTable		Link to table identified by name	
fromColumn		Link to column identified by name	
toTable		Link to table identified by name	
toColumn		Link to column identified by name	

FileInfo

Name	Nullable	Type	Notes
path		Scalar	
name		Scalar	
charSet		Scalar	
fieldDelimiter		Scalar	
fieldEnclosure		Scalar	
recordDelimiter		Scalar	
headerLine		Scalar	
escapeEncoding		Scalar	

© Copyright, WhereScape Software Ltd All rights reserved.

Name	Nullable	Type	Notes
nullEncoding		Scalar	
nonStringNullEncoding		Scalar	

HiveNativeDelimitedRowFormat

Name	Nullable	Type	Notes
delimiter		Scalar	
escapedBy		Scalar	
nullAs		Scalar	
lineTerminator		Scalar	
additionalProperties		Scalar	

HiveNativeFileFormat

Name	Nullable	Type	Notes
storedAs		Scalar	
inputFormatClass		Scalar	
outputFormatClass		Scalar	

HiveNativeSerdeRowFormat

Name	Nullable	Type	Notes
serdeName		Scalar	
serdeProperties		Scalar	

HiveNativeStorage

Name	Nullable	Type	Notes
rowFormat		Scalar	
delimitedRowFormat		HiveNativeDelimitedRowFormat	
serdeRowFormat		HiveNativeSerdeRowFormat	
fileFormat		HiveNativeFileFormat	

HiveNonNativeStorage

Name	Nullable	Type	Notes
storageClass		Scalar	
serdeProperties		Scalar	

HiveStorage

Name	Nullable	Type	Notes
external		Scalar	
externalLocation		Scalar	

© Copyright, WhereScape Software Ltd All rights reserved.

Name	Nullable	Type	Notes
clustered		Scalar	
clusterColumns		Array of column	
clusterSortColumns		Array of column	
clusterBuckets		Scalar	
skew		Scalar	
skewAsDirectories		Scalar	
skewColumns		Array of column	
skewValues		Scalar	
skipFirstRow		Scalar	
native		Scalar	
nativeStorage		**HiveNativeStorage**	
nonNativeStorage		**HiveNonNativeStorage**	
additionalTableProperties		Map of Scalar	

LoadInfo

Name	Nullable	Type	Notes
loadType		enum	
sourceConnection	Y	Link to connection identified by name	
sourceConnectionType		enum	
scriptConnection	Y	Link to connection identified by name	
fileParsed		Scalar	
sourceFile		**FileInfo**	
sqoopSourceFileEnclosureType		enum	
triggerFile		**FileInfo**	
triggerParameterNamePrefix		Scalar	
wait		Scalar	
waitSeconds		Scalar	
waitAction		enum	
archiveFile		**FileInfo**	
triggerArchiveFile		**FileInfo**	
tptHadoopHost		Scalar	
pdwOptions		**PDWLoadOptions**	
selectDistinctValues		Scalar	
sourceSchema		Scalar	
sourceTables		Scalar	
useOverrideSourceColumns		Scalar	
overrideSourceColumns		Scalar	
whereAndGroupByClauses		Scalar	
overrideLoadSQL		Scalar	
fileLoaderOptions		Scalar	

© Copyright, WhereScape Software Ltd All rights reserved.

Name	Nullable	Type	Notes
tempDatabase		Scalar	
tptOptions		TPTLoadOptions	
ssisOptions		SSISLoadOptions	
sqoopOptions		SqoopLoadOptions	

Options

Name	Nullable	Type	Notes
endOfStatement		Scalar	
extendedPropertyScopesById		Map of ExtendedPropertyScope	
extendedPropertyDefinitionsByName		Map of ExtendedPropertyDefinition	
extendedPropertyDefinitions		Array of definitions	

PDWLoadOptions

Name	Nullable	Type	Notes
loadMode		Scalar	
upsertKeyColumns		Scalar	
useStagingDatabase		Scalar	
stagingDatabase		Scalar	
useMultiTransMode		Scalar	

Relation

Name	Nullable	Type	Notes
joinLookup		Scalar	
fk		Scalar	
fks		Array of FK	
ansi		Scalar	
noWarningForUnmatchedBusinessKeys		Scalar	
detailLogForUnmatchedBusinessKeys		Scalar	
useDateColForTrackingChangingDimTable		Scalar	
autoAddUnmatchedBusinessKeys		Scalar	
autoAddProcedure		Scalar	
identifyUnmatchedBusinessKeyMethod		Scalar	
logWhenAutoAdd		Scalar	
currentFlag		Scalar	
trackingDateColumn	Y	column	

© Copyright, WhereScape Software Ltd All rights reserved.

Relationship

Name	Nullable	Type	Notes
joinType		enum	
fromTable		Link to table identified by name	
toTable		Link to table identified by name	
columnJoins		Array of **ColumnJoin**	
primaryJoin		Scalar	

SSISLoadOptions

Name	Nullable	Type	Notes
sourceIdentifierEncapsulation		Scalar	
sourceAlwaysUseDefaultCodePage		Scalar	
destinationAlwaysUseDefaultCodePage		Scalar	
useRowCountLog		Scalar	
loadMode		Scalar	
upsertKeyColumns		Scalar	
rollbackOnTableFailure		Scalar	
useStagingDatabase		Scalar	
stagingDatabase		Scalar	
acquireTableLock		Scalar	
commitInterval		Scalar	
sourceIdentifierCaseConversion		enum	
destinationIdentifierCaseConversion		enum	
sourceCodePage	Y	Scalar	
destinationCodePage	Y	Scalar	
rowsPerBatch	Y	Scalar	

SourceJoinDetails

Name	Nullable	Type	Notes
distinct		Scalar	
groupBy		Scalar	
join		Scalar	
where		Scalar	

SourceTableInfo

Name	Nullable	Type	Notes
sourceTable		Link to table identified by name	
lockingMethod	Y	Scalar	

© Copyright, WhereScape Software Ltd All rights reserved.

SqoopLoadOptions

Name	Nullable	Type	Notes
baseTempDir		Scalar	
genericArguments		Scalar	
toolArguments		Scalar	
splitByColumn	Y	Scalar	

Storage

Name	Nullable	Type	Notes
dataBlockSize		Scalar	
distributionColumns		Array of column	
distributionMethod		Scalar	
fallback		Scalar	
freeSpace		Scalar	
multiSet		Scalar	
optionalCreateClause		Scalar	
organizingColumns		Array of column	
partitionFunction		Scalar	
partitionScheme		Scalar	
partitioned		Scalar	
partitioningColumns		Array of column	
storageIndex	Y	index	
hiveStorage		**HiveStorage**	

TPTLoadOptions

Name	Nullable	Type	Notes
tptRoutine		Scalar	
characterSet		Scalar	
odbcOperatorAttributes		Scalar	
loadOperatorAttributes		Scalar	
jobName		Scalar	
tbuildOptions		Scalar	
tlogviewOptions		Scalar	
sharedMemorySize		Scalar	
sessionsMin		Scalar	
readInstances		Scalar	
writeInstances		Scalar	
sessionsMax	Y	Scalar	

© Copyright, WhereScape Software Ltd All rights reserved.

Table

Name	Nullable	Type	Notes
id		Scalar	
columns		Array of column	
columnsByName		Map of **Column**	
database		Scalar	
dbType		enum	
indexes		Array of Link to index identified by name	
name		Scalar	
objectType		enum	
relations		Array of **Relation**	
relationships		Array of **Relationship**	
schema		Scalar	
shortName		Scalar	
displayName		Scalar	
description		Scalar	
sourceJoinDetails		**SourceJoinDetails**	
storage		**Storage**	
subType		Scalar	
target		Link to target identified by name	
updateProcedure	Y	Link to procedure identified by name	
customProcedure	Y	Link to procedure identified by name	
getFunction	Y	Link to procedure identified by name	
postLoadProcedure	Y	Link to procedure identified by name	
loadScript	Y	Link to host script identified by name	
postExportProcedure	Y	Link to procedure identified by name	
exportScript	Y	Link to host script identified by name	
loadInfo	Y	**LoadInfo**	
exportInfo	Y	**ExportInfo**	
viewInfo	Y	**ViewInfo**	
extendedPropertyValuesByName		Map of Scalar	
sourceMappings	Y	Array of Link to source mapping identified by sourceMappingName	

© Copyright, WhereScape Software Ltd All rights reserved.

TableAdditionalInformationChange

Name	Nullable	Type	Notes
changedKey		Scalar	
changedDescription		Scalar	
expectedValue		Scalar	
actualValue		Scalar	

TableCreation

Name	Nullable	Type	Notes

UpdateProcSettings

Name	Nullable	Type	Notes
batchDateGrain		Scalar	
batchJoinTableColumn		Scalar	
batchProcessingField		Scalar	
changeDetectionEndCurrent		Scalar	
changeDetectionEndExpiring		Scalar	
changeDetectionStartInitial		Scalar	
changeDetectionStartNew		Scalar	
changeDetectionUpdateCurrentRecordsOnly		Scalar	
createStorageIndexAfterCTAS		Scalar	
dateBasedPartitioningInfo		DateBasedPartitioningInfo	
deleteBeforeInsert		Scalar	
deleteBeforeInsertMultiPass		Scalar	
deleteBeforeInsertTruncate		Scalar	
deleteRowsNotInSource		Scalar	
deleteTruncateOption		Scalar	
deleteWhereClause		Scalar	
dssCurrentFlagColumnName		Scalar	
dssEndDateColumnName		Scalar	
dssStartDateColumnName		Scalar	
enableParallelDml		Scalar	
includeExplicitLock		Scalar	
includeInitialLoadInsert		Scalar	
insert		Scalar	
insertHint		Scalar	
insertMinusHint		Scalar	
insertNewRowIdentificationMethodExcept		Scalar	
insertNewRowsOnly		Scalar	

© Copyright, WhereScape Software Ltd All rights reserved.

Name	Nullable	Type	Notes
insertZeroKeyRecord		Scalar	
lockClause		Scalar	
merge		Scalar	
mergeChangedRowsOnly		Scalar	
mergeHint		Scalar	
mergeMinusHint		Scalar	
mergeNewRowIdentificationMethodExcept		Scalar	
minimalLogging		Scalar	
nullSupport		Scalar	
parameters		Array of Scalar	
partitionExchangeTable	Y	Link to table identified by name	
procedureName		Scalar	
processByBatch		Scalar	
selectHint		Scalar	
sourceJoinDetails		**SourceJoinDetails**	
sourceTablesByName		Map of **SourceTableInfo**	
template	Y	template	
update		Scalar	
updateChangeRowIdentificationMethodExcept		Scalar	
updateChangedRowsOnly		Scalar	
updateCurrentRowsOnly		Scalar	DEPRECATED: Use updateChangedRowsOnly instead
updateHint		Scalar	
updateMinusHint		Scalar	
warnOnDelete		Scalar	

VanillaSettings

Name	Nullable	Type	Notes
template	Y	template	

ViewInfo

Name	Nullable	Type	Notes
viewType		enum	
distinct		Scalar	
whereClause		Scalar	
userDefined		Scalar	
userDefinition		Scalar	

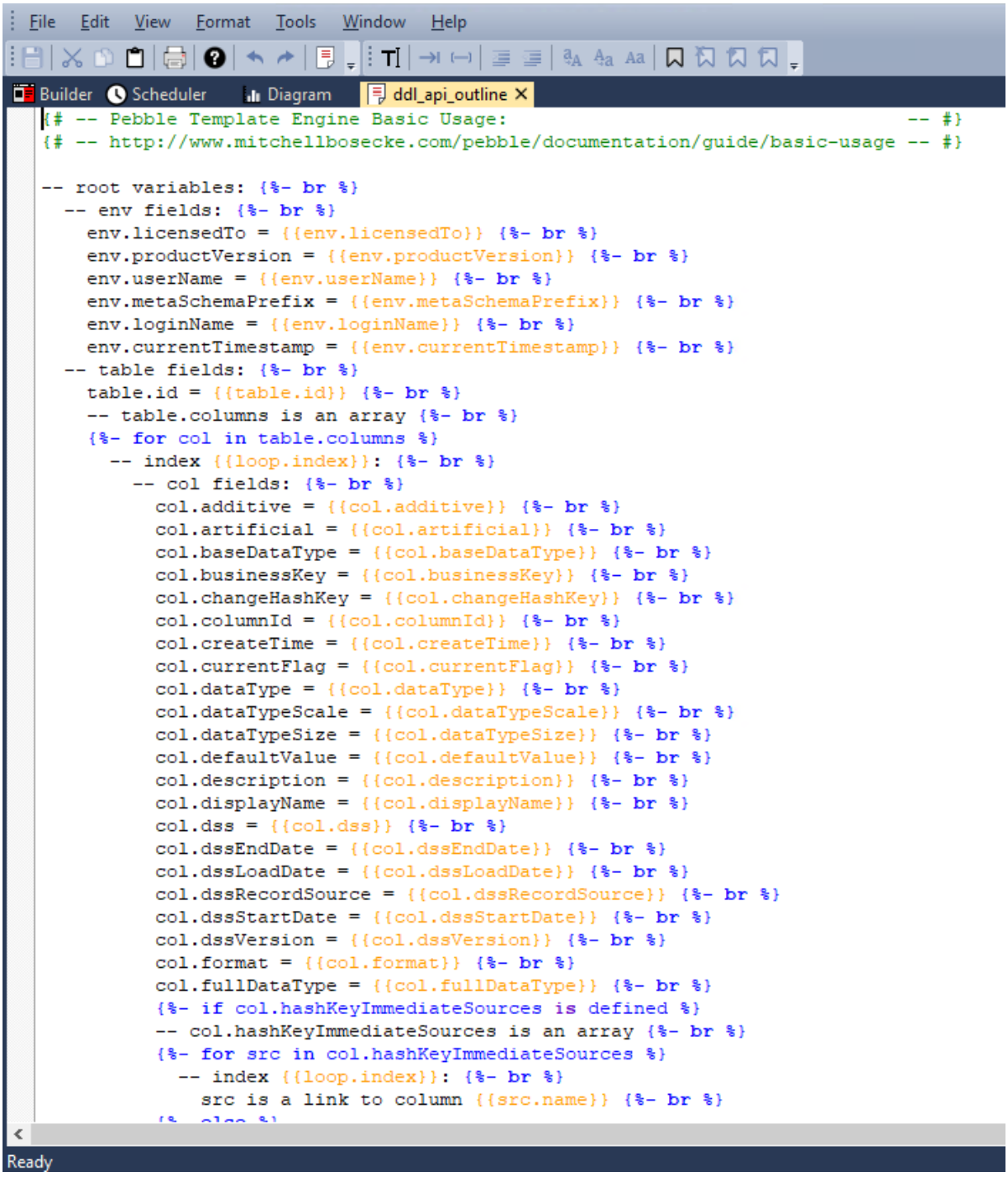
© Copyright, WhereScape Software Ltd All rights reserved.

Template Editor

Right-click a template and select **Edit the Template** or **View Template** to open the Template editor.

Similar to the procedure and script editors, the template editor includes the following features:

- Pebble syntax highlighting
- improved find feature (repeated find, up and down, etc.)
- toggle the display of line numbering
- a status bar that includes line number, line length, selected text length, etc.
- a context sensitive toolbar
- tab to space conversion and vice versa
- uppercase, lowercase, titlecase text conversion command
- increase/decrease test indent command
- bookmark support



© Copyright, WhereScape Software Ltd All rights reserved.

Notes

Editing or deleting Locked for Edit or opened Templates

- Templates cannot be deleted if they are Locked for Edit or checked out by another user.
- Saving the template cannot be performed if the template is Locked for Edit by other users after the edit window has been opened.
- Templates cannot be deleted or modified after the editor window has been opened, unless the Edit Lock has been released. Edit Locks can be released by any user in the Template Properties screen.
- To prevent updates, deletes and modifications to certain Templates, it is best to use the Check Out functionality instead. For more information about this functionality, refer to **Object Check Outs and Check Ins** for details.

Evaluating an API Outline Template

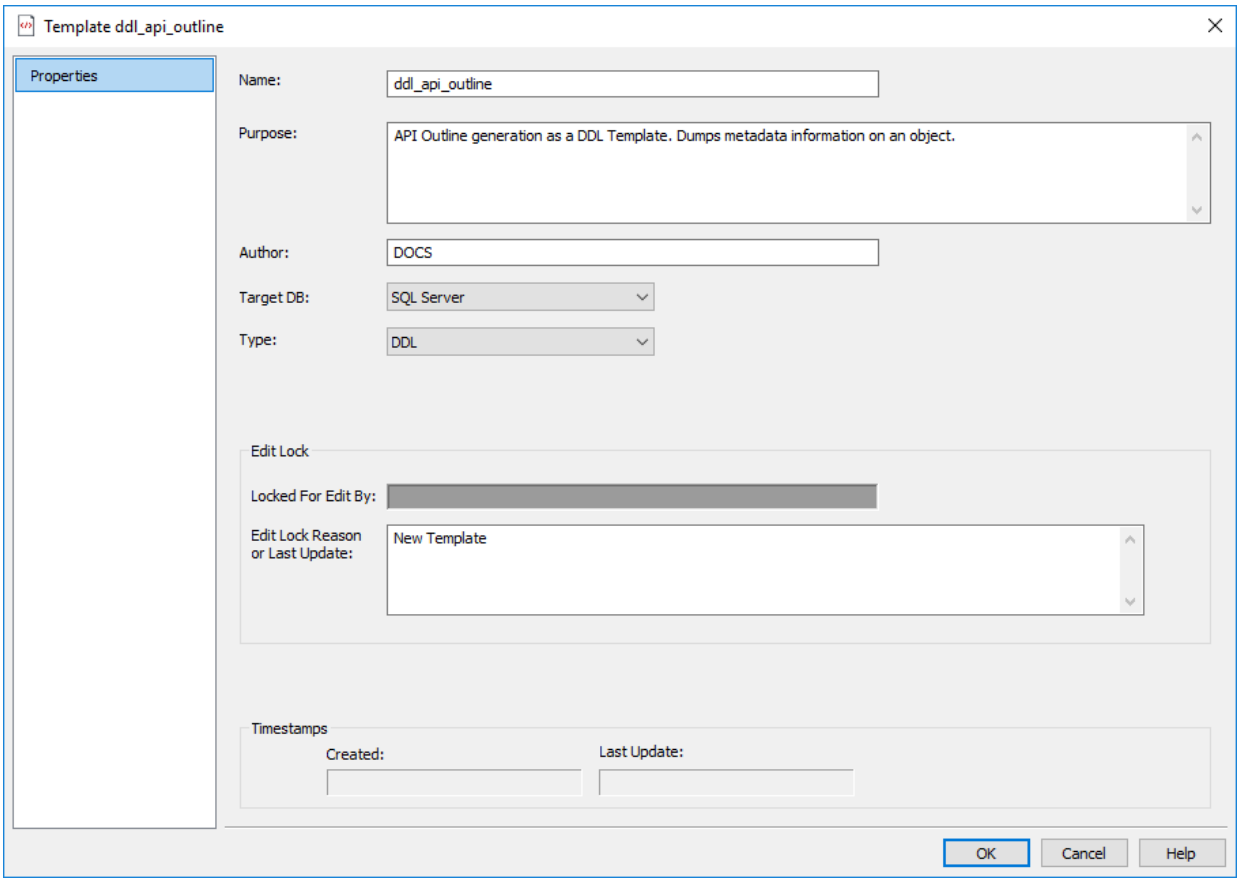
An API Outline Template is available to output all object properties relevant to the current object. Upon evaluation of this template, the status of each property is generated and printed to the script or procedure file.

Note

Template evaluation usually generates a script or procedure file, but the API Outline Template generates plain text. The output of this template is intended to be viewed or copied to a text file, it cannot be executed as a script.

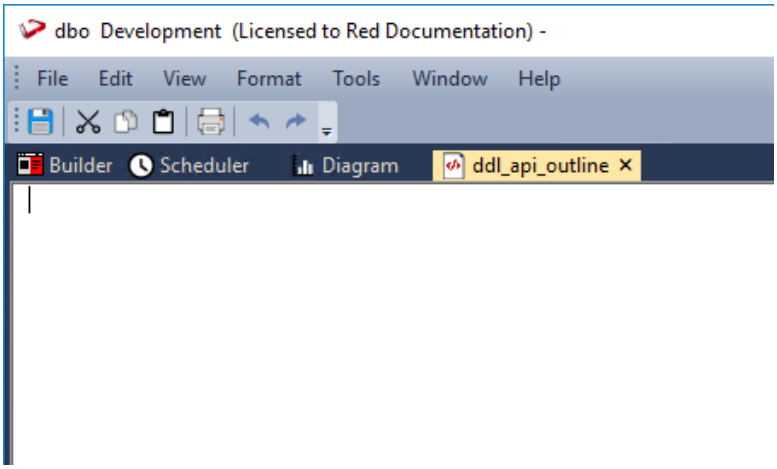
To evaluate an API Outline Template:

1. Create a **new template**. The template can normally be of any type, but in this example we'll use the DDL template type as viewing the evaluation of DDL templates is simple.
   Set **Target DB** to your source connection database type. In this example, we will set the Target DB to SQL Server because the load table we are evaluating is stored on a SQL Server connection.

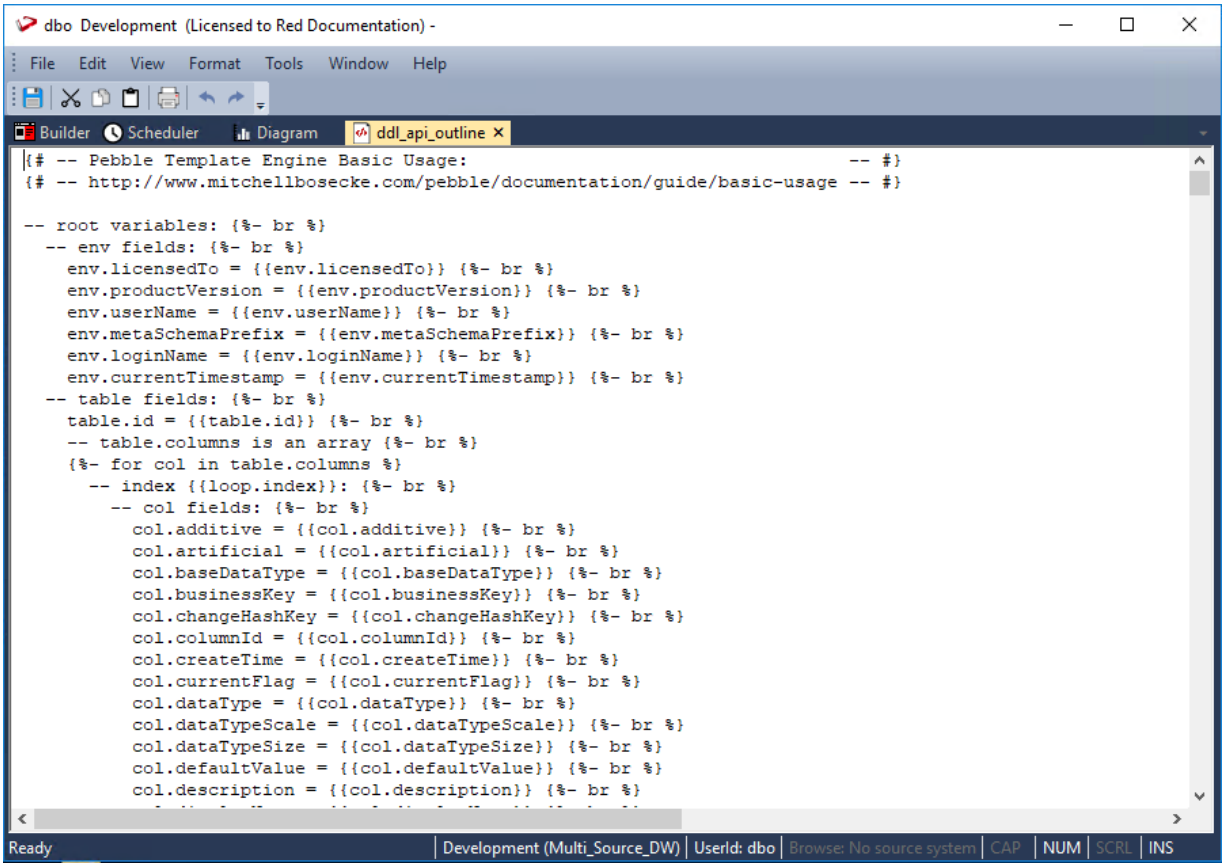


© Copyright, WhereScape Software Ltd All rights reserved.

2. Open the template in the Template Editor.



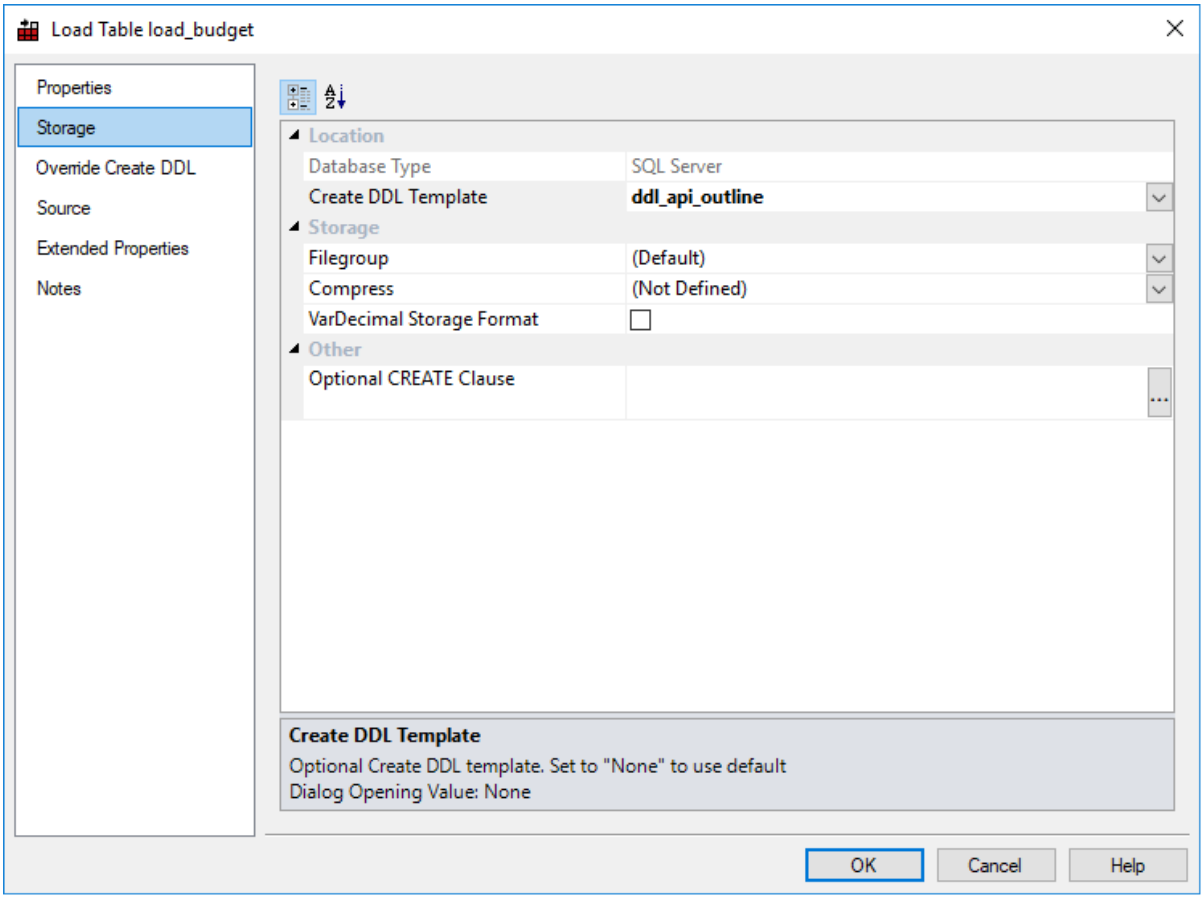
3. Click **Tools > Create API Example Outline**. The API Example Outline text is added to the blank template.



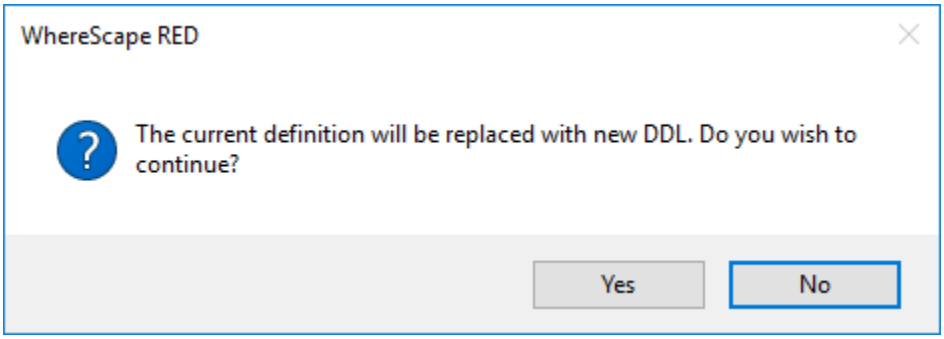
4. **Save** and **close** the template.
5. Open the Properties dialog for the **Load Table** you wish to evaluate.

© Copyright, WhereScape Software Ltd All rights reserved.

6. In the **Storage** tab, select the template you created in the **Create DDL Template** drop-down box.

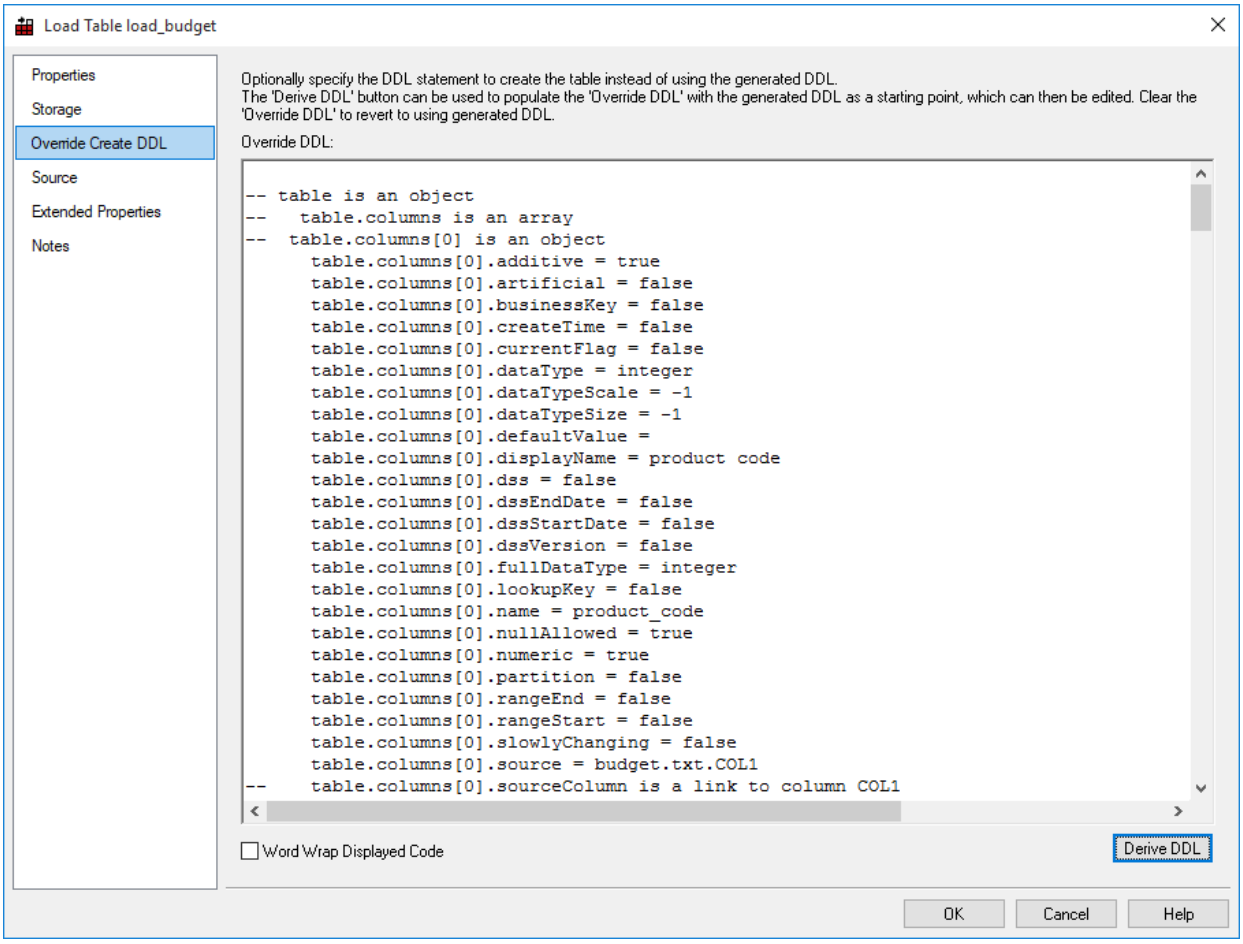


7. Open the **Override Create DDL** tab. If the **Override DDL** field is populated with a custom DDL statement, copy and paste this statement to a text file for backup purposes.
8. Click the **Derive DDL** button. A warning dialog box displays informing you that the association of the DDL template with this table will be saved to the metadata, click **OK**.



© Copyright, WhereScape Software Ltd All rights reserved.

9. The results of the API Outline Template are printed to the **Override DDL** text box. Cut or copy this text to a text editor and save as a text file for reference purposes.



10. Click **Cancel** in the Load table Properties dialog.

Note

Ensure the Load table properties are returned to their previous state. The default value for **Storage > Create DDL Template** is **None** and the **Override Create DDL > Override DDL** field is left blank to use the automatically generated DDL statement or ensure your custom DDL statement remains.

© Copyright, WhereScape Software Ltd All rights reserved.

Pebble Syntax

Notes

- Pebble syntaxes documented in this section are common between WhereScape products, but some syntaxes are unique to WhereScape 3D and WhereScape RED.
- It is advisable that you do not copy contents of a Pebble template from one WhereScape product to another.

Controls

Controls	Description	Example

{%...%}	This is a tag for control flow. It should contain a tag and may be additional parameters, as shown in the example.	`{% if ${condition} %}`
		`${content}`
		`{% else %}`
		`${content}`
		`{% endif %}`

{%-...-%}	This is a tag for control flow. It should contain a tag and may be additional parameters, as shown in the example. If there is a **-** at the start or at the end, this will trim the leading or trailing whitespace respectively.	`{%- if ${condition} -%}`
		`${content}`
		`{%- else -%}`
		`${content}`
		`{%- endif -%}`

{{...}}	This tag is for expressions. It should contain an expression that will be outputted, as shown in the example.	`{{ max(5, 3) }}`
{{-...-}}	This tag is for expressions. It should contain an expression that will be outputted, as shown in the example. If there is a **-** at the start or at the end, this will trim the leading or trailing whitespace respectively.	`{{- range(3, 6) -}}`
{#...#}	This tag is for comments.	`{# this is a comment. #}`
{#-...-#}	This tag is for comments. If there is a **-** at the start or at the end, this will trim the leading or trailing whitespace respectively.	`{#- No whitespace on the left #}`

© Copyright, WhereScape Software Ltd All rights reserved.

Pebble Template Tags

autoescape

The autoescape tag can be used to temporarily disable/re-enable the autoescaper as well as change the escaping strategy for a portion of the template.

Example

```
{% autoescape "js" %}
${content}
{% endautoescape %}
```

block

A section that can be overridden by a child template.

Template

```
{% block "${blockName}" %}
${content}
{% endblock %}
```

extends

The extends tag is used to declare a parent template. It should be the very first tag used in a child template and a child template can only extend up to one parent template.

Template

```
{% extends "${templateName}" %}
```

filter

The filter tag allows you to apply a filter to a large chunk of template.

Template

```
{% filter ${filter} %}
${content}
{% endfilter %}
```

Template 2

```
{% filter ${filter1} | ${filter2} %}
${content}
{% endfilter %}
```

for

The for tag is used to iterate through a collection or map.

Template

```
{% for ${i} in ${collection} %}
${content}
{% endfor %}
```

© Copyright, WhereScape Software Ltd All rights reserved.

for else

The for tag is used to iterate through a collection or map with a convenient way to check for emptiness.

Template

```
{% for ${i} in ${collection} %}
${content}
{% else %}
${content}
{% endfor %}
```

if

The if tag allows you to designate a chunk of content as conditional depending on the result of an expression.

Template

```
{% if ${condition} %}
${content}
{% endif %}
```

if else

The if tag allows you to designate a chunk of content as conditional depending on the result of an expression.

Template

```
{% if ${condition} %}
${content}
{% else %}
${content}
{% endif %}
```

if elseif

The if tag allows you to designate a chunk of content as conditional depending on the result of an expression.

emplate

```
{% if ${condition} %}
${content}
{% elseif %}
${content}
{% else %}
${content}
{% endif %}
```

import

The import tag allows you to use macros defined in another template.

Template

```
{% import "${templateName}" %}
```

© Copyright, WhereScape Software Ltd All rights reserved.

include

The include tag allows you to insert the rendered output of another template directly into the current template. The included template will have access to the same variables that the current template does.

Template

```
{% include "${templateName}" %}
```

include with

The include tag allows you to insert the rendered output of another template directly into the current template. The included template will have access to the same variables that the current template does with additional variables.

Template

```
{% include "${templateName}" with { ${mapOfVariables} } %}
```

macro

The macro tag allows you to create a chunk of reusable and dynamic content. The macro can be called multiple times in the current template or even from another template with the help of the import tag.

Example

```
{% macro concat(left, right) %}
{{ left }} {{ right }}
{% endmacro %}
```

parallel

The parallel tag allows you to designate a chunk of content to be rendered using a new thread.

Template

```
{% parallel %}
${content}
{% endparallel %}
```

set

The set tag allows you to define a variable in the current context, whether it currently exists or not.

Will initialize the variable or override the current value in the variable.

Template

```
{% set ${variable} = ${value} %}
```

verbatim

The verbatim tag allows you to write Pebble syntax that won't be parsed.

Example

```
{% verbatim %}
{% if not varA is defined %}
{% set varA = "Abc" %}
{% endif %}
{% endverbatim %}
```

© Copyright, WhereScape Software Ltd All rights reserved.

br

The br tag adds a line break token at the current position.

Use an additional parameter to change the type of line break: lf (Default), crlf, other "token".

Template

```
{% br %}
{% br lf %}
{% br crlf %}
{% br other "|" %}
```

counter

The counter tag outputs an integer value that is incremented by one for each use.

Use the parameter local to make the counter increments per used template.

Example

```
{% counter %} pencil
{% counter %} pens
```

Output

```
1 pencil
2 pens
```

Example 2

```
Template "Other": {% counter local %}
Main template: {% counter local %}{% include "Other" %}
```

Outputs

```
11
```

declare

The declare tag allows you to define a variable in the current context if it does not exist. If the variable does exist it will not be changed.

Note

The following are the same:

```
{% declare encapsulation = true %}{% set encapsulation = encapsulation | default(true) %}
```

Example

```
{% set varA = "aa" %}
{% declare varA = "bb" %}
{% declare varB = "cc" %}
{{ varA }} {{ varB }}
```

Outputs

```
aa cc
```

© Copyright, WhereScape Software Ltd All rights reserved.

error

The error tag allows you to fail generating the template with an error message, e.g. if something unexpected occurs.

Example

```
{% if not requiredVar is defined %}
{% error "Something terrible has occurred" %}
```

indent

The indent tag adds one or more 4-space indents at the current position.

Example

```
{% indent %}
```

Example 2

```
{% indent 3 %}
```

from

The from tag is used to iterate a subset of a collection defined by the where condition.

Template

```
{% from ${collection} as ${var} where ${test} %}
${content}
{% endfrom %}
```

Example

```
{% from range(1,10) as num where num is even %}
{{- num }}{% br %}
{% endfrom %}
```

Output

```
2
4
6
8
10
```

from else

The from tag is used to iterate a subset of a collection defined by the where condition with a convenient way to check for emptiness.

Template

```
{% from ${collection} as ${var} where ${test} %}
${content}
{% else %}
${content}
{% endfrom %}
```

© Copyright, WhereScape Software Ltd All rights reserved.

Example

```
{% from [] as num where num is even -%}
Even numbers{% br %}
{% else -%}
Empty collection{% endfrom %}
```

Output

```
Empty collection
```

list add

The list with add tag is used to append a value to a list.

Example

```
{% set alphabet = ["a", "b", "c"] %}
{{ alphabet }}{% br %}
{% list alphabet add "d" -%}
{{ alphabet }}
```

Outputs

```
[a, b, c]
[a, b, c, d]
```

list adds

The list with add tag is used to append one or more values to a list.

Example

```
{% set alphabet = ["a", "b", "c"] %}
{{ alphabet }}{% br %}
{% list alphabet add "d", "e", "f" -%}
{{ alphabet }}
```

Outputs

```
[a, b, c]
[a, b, c, d, e, f]
```

list remove

The list with remove tag is used to remove a value from a list.

Example

```
{% set alphabet = ["a", "b", "c", "d", "e", "f"] %}
{{ alphabet }}{% br %}
{% list alphabet remove "d" -%}
{{ alphabet }}
```

Outputs

```
[a, b, c, d, e, f]
[a, b, c, e, f]
```

© Copyright, WhereScape Software Ltd All rights reserved.

list removes

The list with remove tag is used to remove one or more values from a list.

Example

```
{% set alphabet = ["a", "b", "c", "d", "e", "f"] %}
{{ alphabet }}{% br %}
{% list alphabet remove "d", "e", "f" -%}
{{ alphabet }}
```

Outputs

```
[a, b, c, d, e, f]
[a, b, c]
```

map put

The map put tag is used to set a key-value pair into a map, whether it currently exists or not.

Template

```
{% map ${mapVar} put ${key} = ${value} %}
```

map remove

The map remove tag is used to remove a key from a map.

Template

```
{% map ${mapVar} remove ${key} %}
```

placebottom

The placebottom tag allows defining a block in the template that will be placed at the end of the output. The priority argument defines the order of multiple place bottom blocks.

Template

```
{% placebottom ${priority} %}
${content}
{% endplacebottom %}
```

get

The get tag defines a variable as a subset of a collection based on the given condition.

Example

```
{% get evenNums from range(1,10) as num where num is even %}
{{ evenNums }}
```

Output

```
[2, 4, 6, 8, 10]
```

© Copyright, WhereScape Software Ltd All rights reserved.

WhereScape RED Specific Tag

fetch

The fetch tag is used to manually load attributes of objects that which aren't loaded automatically, eg. for an alternate table.

Example

```
{% for col in table.columns %}
 {% fetch col.sourceTable %}
 {% set srcTab = col.sourceTable %}
 {% for srcCol in srcTab.columns  %}
{{ srcCol.source }}{% br %}
 {% endfor %}
{% endfor %}
```

Pebble Template Functions

block

The block function is used to render the contents of a block tag more than once.

Example

```
{% block "testBlock" %}Some content
{% endblock %}
{{ block("testBlock") }}
```

Output

```
Some contentSome content
```

max

The max function will return the largest of it's numerical arguments.

Example

```
{{ max(5, 3) }}
```

Output

```
5
```

min

The min function will return the smallest of it's numerical arguments.

Example

```
{{ min(5, 3) }}
```

```
3
```

© Copyright, WhereScape Software Ltd All rights reserved.

parent

The parent function is used inside of a block to render the content that the parent template would have rendered inside of the block had the current template not overriden it.

Template

```
{% block "${blockName}" %}
{{ parent() }}
{% endblock %}
```

range

The range function will return a list containing an arithmetic progression of numbers.

Example

```
{{ range(3, 6) }}
```

Output

```
[3, 4, 5, 6]
```

range with step

The range function will return a list containing an arithmetic progression of numbers.

Example

```
{{ range(2, 10, 2) }}
```

Output

```
[2, 4, 6, 8, 10]
```

bool

The bool function will return a boolean from parsing a value.

Example

```
{{ bool("T") }}
```

Output

```
true
```

Example 2

```
{{ bool(0) }}
```

Output

```
false
```

codepoint

The codepoint function will return the character for the specified Unicode code point.

Example

```
{{ codepoint(34) }}
```

© Copyright, WhereScape Software Ltd All rights reserved.

Output

```
"
```

combine lists

The `combineLists` function will return a merged list of the two arguments.

Example

```
{% set listA = [ "a", "b", "c" ] %}
{% set listB = [ "d", "c", "e" ] %}
{{ combineLists(false, listA, listB) }}
{{ combineLists(true, listA, listB) }}
```

Output

```
[a, b, c, d, c, e][a, b, c, d, e]
```

double

The double function will return an int for the provided Number argument or attempt to parse a String into a double.

Example

```
{{ double(3.145) }}
```

Output

```
3.145
```

Example 2

```
{{ double("3.145") }}
```

Output

```
3.145
```

int

The int function will return an int for the provided Number argument or attempt to parse a String into an int.

Example

```
{{ int(3.145) }}
```

Output

```
3
```

Example 2

```
{{ int("3.145") }}
```

Output

```
3
```

© Copyright, WhereScape Software Ltd All rights reserved.

replace start

The `replaceStart` function will return a String where **str** will be checked if it starts with **match** and if so be

Example

```
{{ replaceStart("prefix", "pre", "suf", false) }}
```

Output

```
suffix
```

replace end

The `replaceEnd` function will return a String where **str** will be checked if it ends with **match** and if so be replaced with **replace**.

Example

```
{{ replaceEnd("completion", "ion", "ed", false) }}
```

Output

```
completed
```

str is equals

The strIsEquals function will return a boolean for whether the provided String arguments are equal. Safely handles null values.

Example

```
{{ strIsEquals("a", "A", true) }}
```

Output

```
true
```

Example 2

```
{{ strIsEquals("a", null) }}
```

Output

```
false
```

starts with ignore case

The `startsWithIgnoreCase` function will return a boolean for whether the provided **str** begins with **match** while ignoring casing.

Example

```
{{ startsWithIgnoreCase("SuperMan", "super") }}
```

Output

```
true
```

© Copyright, WhereScape Software Ltd All rights reserved.

ends with ignore case

The `endsWithIgnoreCase` function will return a boolean for whether the provided **str** ends with **match** while ignoring casing.

Example

```
{{ endsWithIgnoreCase("SuperMan", "man") }}
```

Output

```
true
```

Pebble Template Filters

abbreviate

The abbreviate filter will abbreviate a string using an ellipsis. It takes one argument which is the max width of the desired output including the length of the ellipsis.

Example

```
{{ "This is a sentence" | abbreviate(7) }}
```

Output

```
This...
```

abs

The abs filter is used to obtain the absolute value.

Example

```
{{ -5 | abs }}
```

Output

```
5
```

capitalize

The capitalize filter will capitalize the first letter of the string.

Example

```
{{ "this is a sentence" | capitalize }}
```

Output

```
This is a sentence
```

© Copyright, WhereScape Software Ltd All rights reserved.

date

The date filter is used to format an existing `java.util.Date` object. The filter will construct a `java.text.SimpleDateFormat` using the provided pattern and then use this newly `created SimpleDateFormat` to format the provided Date object.

The alternative way to use this filter is to use it on a string but then provide two arguments. The first is the desired pattern for the output and the second is the existing format used to parse the input string into a `java.util.Date` object.

Example : If todayDate is a java.util.Date object

```
{{ todayDate | date("yyyy-MM-dd") }}
```

Output

```
2017-12-25
```

Example 2

```
{{ "December 25, 2017" | date("yyyy-MM-dd", existingFormat="MMMM dd, yyyy") }}
```

Output

```
2017-12-25
```

default

The default filter will render a default value if and only if the object being filtered is empty. A variable is empty if it is null, an empty string, an empty collection, or an empty map.

Example

```
{%- set str = "" -%}
{{ str | default("NonEmpty") }}
```

Output

```
NonEmpty
```

escape

The escape filter will turn special characters into safe character references in order to avoid XSS vulnerabilities.

Example

```
{{ "<content>" | escape }}
```

Output

```
<content>
```

© Copyright, WhereScape Software Ltd All rights reserved.

first

The first filter will return the first item of a collection, or the first letter of a string.

Example

```
{{ "Some letters" | first }}
```

Output

```
S
```

Example 2

```
{{ [ "b", "a", "c" ] | first }}
```

Output

```
b
```

join

The join filter will concatenate all items of a collection into a string. An optional argument can be given to be

Example

```
{{ [ "This", "is", "a", "sentence" ] | join }}
```

Output

```
This is a sentence
```

Example 2

```
{{ [ "This", "is", "a", "sentence" ] | join(" ") }}
```

Output

```
This is a sentence
```

last

The last filter will return the last item of a collection, or the last letter of a string.

Example

```
{{ "Some letters" | last }}
```

Output

```
s
```

Example 2

```
{{ [ "b", "a", "c" ] | last }}
```

Output

```
c
```

© Copyright, WhereScape Software Ltd All rights reserved.

length

The length filter returns the number of items of collection, map or the length of a string.

Example

```
{{ "Some letters" | length }}
```

Output

```
12
```

Example 2

```
{{ [ "b", "a", "c" ] | length }}
```

Output

```
3
```

lower

The lower filter makes an entire string lower case.

Example

```
{{ "THIS IS A SENTENCE" | lower }}
```

Output

```
this is a sentence
```

merge

The merge filter returns a new map, list or array that is the result of merging two maps, lists or arrays together.

Note

Keys in the second map will overwrite the first map keys.

Example

```
{%- set mapa = { "a": "aa", "b": "bb", "c": "cc" } -%}
{%- set mapb = { "c": "ccc", "d": "ddd", "e": "eee" } -%}
{{ mapa | merge(mapb) }}
```

Output

```
{a=aa, b=bb, c=ccc, d=ddd, e=eee}
```

numberformat

The `numberformat` filter is used to format a decimal number. Behind the scenes it uses `java.text.DecimalFormat`.

Example

```
{{ 3.14678 | numberformat("#.##") }}
```

Output

```
3.15
```

© Copyright, WhereScape Software Ltd All rights reserved.

raw

The raw filter prevents the output of an expression from being escaped by the autoescaper. The raw filter must be the very last operation performed within the expression otherwise the autoescaper will deem the expression as potentially unsafe and escape it regardless.

Note

If the raw filter is not the last operation performed then the expression will be escaped.

Example

```
{{ "<content>" | raw }}
```

Output

```
<content>
```

replace

The replace filter returns a string that is the result of replacing any substrings matching keys in a map with the respective values.

Example

```
{{ "This is a sentence" | replace({ "i": "eye" }) }}
```

Output

```
Theyes eyes a sentence
```

rsort

The rsort filter will sort a list in reversed order. The items of the list must implement Comparable.

Example

```
{{ [ "b", "a", "c" ] | rsort }}
```

Output

```
[c, b, a]
```

slice

The slice filter returns a portion of a list, array, or string. The indexing starts at 0.

xample

```
{{ "This is a sentence" | slice(5, 9) }}
```

Output

```
is a
```

Example 2

```
{{ [ "b", "a", "c" ] | slice(1, 2) }}
```

Output

```
[a[
```

© Copyright, WhereScape Software Ltd All rights reserved.

sort

The sort filter will sort a list. The items of the list must implement Comparable.

Example

```
{{ [ "b", "a", "c" ] | sort }}
```

Output

```
[a, b, c[
```

title

The title filter will capitalize the first letter of each word.

Example

```
{{ "This is a sentence" | title }}
```

Output

```
This Is A Sentence
```

trim

The trim filter is used to trim whitespace off the beginning and end of a string.

Example

```
{{ " Some letters " | trim }}
```

Output

```
Some letters
```

upper

The upper filter makes an entire string upper case.

Example

```
{{ "this is a sentence" | upper }}
```

Output

```
THIS IS A SENTENCE
```

urlencode

The urlencode filter translates a string into `application/x-www-form-urlencoded` format using the `UTF-8` encoding scheme.

Example

```
{{ "list=[a,b,c]" | urlencode }}
```

Output

```
list%3D%5Ba%2Cb%2Cc%5D
```

© Copyright, WhereScape Software Ltd All rights reserved.

lines

The lines filter will split a string by its line breaks with each line becoming an element in a list.

Example

```
{%- set aStr = "Something
Is
On
Lots
Of
Lines" -%}
{{ aStr | lines }}
```

Output

```
[Something, Is, On, Lots, Of, Lines]
```

Pebble Template Tests

defined

The defined test checks if a variable exists.

Example

```
{%if someVar is defined%}    ...
{%endif %}
```

empty

The empty test checks if a variable is empty. A variable is empty if it is null, an empty string, an empty collection, or an empty map.

Example

```
{% if "" is empty %}
...
{% endif  %}
```

even

The even test checks if an integer is even.

Example

```
{% if 2 is even %}
...
{% endif  %}
```

© Copyright, WhereScape Software Ltd All rights reserved.

iterable

The iterable test checks if a variable implements `java.lang.Iterable`.

Example

```
{% if [ "a", "b", "c" ] is iterable %}
...
{% endif  %}
```

map

The map test checks if a variable is an instance of a map.

Example

```
{% if { "a": "aa", "b": "bb" } is map %}
...
{% endif  %}
```

null

The null test checks if a variable is null.

Example

```
{% if someVar is null %}
...
{% endif  %}
```

odd

The odd test checks if an integer is odd.

Example

```
{% if "" is odd %}
...
{% endif  %}
```

boolean

The boolean test checks if the input is a Boolean or viable to be casted as a Boolean.

Example

```
{% if "yes" is boolean %}   ...
{% endif  %}
```

Example 2

```
{% if 1 is boolean %}   ...
{% endif  %}
```

© Copyright, WhereScape Software Ltd All rights reserved.

instanceof

The instanceof test checks if the input is of the specified class.

Example

```
{% if "a" is instanceof ("String") %}   ...
{% endif  %}
```

Example 2

```
{% if "a" is instanceof ("java.lang.String") %}   ...
{% endif  %}
```

number

The number test checks if the input is a Number or viable to be casted as a Number. The input can be either a numeric value or a String.

Example

```
{% if3.14 is number %}   ...
{% endif  %}
```

Example 2

```
{% if "3.14"  is number %}   ...
{% endif  %}
```

string

The string test checks if the input is a String.

Example

```
{% if "a" is string %}...
{% endif  %}
```

© Copyright, WhereScape Software Ltd All rights reserved.

Pebble Template Variables

StringUtil

Provides common string function support.

Function	Description	Datatype	

capitalize(String str)	Capitalizes a String, Changing the first character to title case.	String	**Example**
			`{{ StringUtils.capitalize( "word") }}`
			Output
			`Word`

compare(String str1, String str2)	Compare two Strings lexicographically.
	• Returns -1 when str1 is less than str2.
	• Returns 0 when str1 and str2 are equal.
	• Returns 1 when str1 is greater than str2.
		int	**Example**
			`{% if (StringUtils.compare("lower", "upper") > 0) %}`
			…
			`{% endif %}`

compare(String str1, String str2, boolean nullIsLess)	Compare two Strings lexicographically.
	• Returns -1 when str1 is less than str2.
	• Returns 0 when str1 and str2 are equal.
	• Returns 1 when str1 is greater than str2.
		int	**Example**
			`{% if (StringUtils.compare("lower", "upper", false) > 0) %}`
			…
			`{% endif %}`

compareIgnoreCase(String str1, String str2)	Compare two Strings lexicographically, ignoring case differences.
	• Returns -1 when str1 is less than str2.
	• Returns 0 when str1 and str2 are equal.
	• Returns 1 when str1 is greater than str2.
		int	**Example**
			`{% if (StringUtils.compareIgnoreCase("lower", "upper") > 0) %}`
			…
			`{% endif %}`

compareIgnoreCase(String str1, String str2, boolean nullIsLess)	Compare two Strings lexicographically, ignoring case differences.
	• Returns -1 when str1 is less than str2.
	• Returns 0 when str1 and str2 are equal.
	• Returns 1 when str1 is greater than str2.
		int	**Example**
			`{% if (StringUtils.compareIgnoreCase("lower", "upper", false) > 0) %}`
			…
			`{% endif %}`

countMatches(String str, String pattern)	Returns the number of times the pattern appears in the string	int	**Example** `{{ StringUtils.countMatches("AaaaaaA", "a") }}`

© Copyright, WhereScape Software Ltd All rights reserved.

Function	Description	Datatype	
			Output 5
indexOf(String str, String search)	Returns the index (Starts with zero) within the String of the first occurrence of the specified search String.	int	**Example** {{ StringUtils.indexOf("abcabcabc", "c") }} **Output** 2
indexOf(String str, String search, int startPos)	Returns the index (Starts with zero) within the String of the first occurrence of the specified search String starting from the provided zero based index	int	**Example** {{ StringUtils.indexOf("abcabcabc", "c", int(4)) }} **Output** 5
indexOfIgnoreCase(String str, String search)	Returns the case insensitive index (Starts with zero) within the String of the first occurrence of the specified search String.	int	**Example** {{ StringUtils.indexOfIgnoreCase("ABCABCABC", "c") }} **Output** 2
indexOfIgnoreCase(String str, String search, int startPos)	Returns the case insensitive index (Starts with zero) within the String of the first occurrence of the specified search String starting from the provided zero based index.	int	**Example** {{ StringUtils.indexOfIgnoreCase("ABCABCABC", "c", int(4)) }} 5
isAllLowerCase(String str)	Checks if the String contains only lowercase characters.	boolean	**Example** {{ StringUtils.isAllLowerCase("letters") }} {% br %} {{ StringUtils.isAllLowerCase("leTTers") }} **Output** true false
isAllUpperCase(String str)	Checks if the String contains only uppercase characters.	boolean	**Example**

© Copyright, WhereScape Software Ltd All rights reserved.

Function	Description	Datatype	
			{{ StringUtils.isAllUpperCase("LETTERS") }} {% br %} {{ StringUtils.isAllUpperCase("leTTers") }} **Output** true false
isAlpha(String str)	Checks if the String contains only unicode letters.	boolean	**Example** {{ StringUtils.isAlpha("letters") }}{% br %} {{ StringUtils.isAlpha("Num1") }} **Output** true false
isAlphanumeric(String str)	Checks if the String contains only unicode letters or digits.	boolean	**Example** {{ StringUtils.isAlphanumeric("Number3") }}{% br %} {{ StringUtils.isAlphanumeric("Number 3") }} **Output** true false
isAlphanumericSpace(String str)	Checks if the String contains only unicode letters, digits or space.	boolean	**Example** {{ StringUtils.isAlphanumericSpace("Number 3") }} **Output** true
isAlphaSpace(String str)	Checks if the string contains only unicode letters or space.	boolean	**Example** {{ StringUtils.isAlphaSpace("next letter") }}{% br %}

© Copyright, WhereScape Software Ltd All rights reserved.

Function	Description	Datatype	
			{{ StringUtils.isAlphaSpace("Number 3") }} **Output** true false
isNumeric(String str)	Checks if the String contains only unicode digits.	boolean	**Example** {{ StringUtils.isNumeric("123") }}{% br %} {{ StringUtils.isNumeric("Number2") }} **Output** true false
isNumericSpace(String str)	Checks if the String contains only unicode digits or space.	boolean	**Example** {{ StringUtils.isNumericSpace("1 2 3") }}{% br %} {{ StringUtils.isNumericSpace("Johnny 5") }} **Output** true false
isWhitespace(String str)	Checks if the String contains only whitespace.	boolean	**Example** {{ StringUtils.isWhitespace(" ") }}{% br %} {{ StringUtils.isWhitespace(" letter ") }} **Output** true false
lastIndexOf(String str, String search)	Returns the index (Starts with zero) within the String of the last occurrence of the specified search String.		**Example** {{ StringUtils.lastIndexOf("abcabcabc", "a") }}

© Copyright, WhereScape Software Ltd All rights reserved.

Function	Description	Datatype	
			Output `6`

lastIndexOf(String str, String search, int startPos)	Returns the index (Starts with zero) within the String of the last occurrence of the specified search String starting from the provided zero based index.		**Example**
			`{{ StringUtils.lastIndexOf("abcabcabc", "a", int(2)) }}`
			Output
			`0`

lastIndexOfIgnoreCase (String str, String search)	Returns the case insensitive index (Starts with zero) within the String of the last occurrence of the specified search String.	int	**Example**
			`{{ StringUtils.lastIndexOfIgnoreCase("ABCABCABC", "a") }}`
			Output
			`6`

lastIndexOf(String str, String search, int startPos)	Returns the case insensitive index (Starts with zero) within the String of the last occurrence of the specified search String starting from the provided zero based index.	int	**Example**
			`{{ StringUtils.lastIndexOfIgnoreCase("ABCABCABC", "a", int(2)) }}`
			Output
			`0`

leftPad(String str, int size)	Left pad a String with spaces.	String	**Example**
			`T{{ StringUtils.leftPad("1", int(5) }}T`
			Output
			`T 1T`

repeat(String str, int repeat)	Repeat a String repeat times to form a new String.	String.	**Example**
			`{{ StringUtils.repeat("haha", int(5)) }}`
			Output
			`hahahahahahahahahaha`

repeat(String str, String separator, int repeat)	Repeat a String repeat times to form a new String, with a String separator injected each time.		**Example**
			`{{ StringUtils.repeat("haha", " ", int(5)) }}`
			Output
			`haha haha haha haha haha`

© Copyright, WhereScape Software Ltd All rights reserved.

Function	Description	Datatype	

reverse(String str)	Reverses a String	String	**Example**
			`{{ StringUtils.reverse("abcde") }}`
			Output
			`edcba`

rightPad(String str, int size)	Right pad a String with spaces.	String	**Example**
			`T{{ StringUtils.rightPad("1", int(5) }}T`
			Output
			`T1 T`

splitByCharacterType(String str)	Splits a String into an array of String with each element containing a substring based on the character type.
	Character types include: lower case, upper case, numbers, whitespace, math operators
		String[]	**Example**
			`{% set array = StringUtils.splitByCharacterType("AbcdEFG%++-/3432 ") %}`
			`Split content:{%- br -%}`
			`{%- for item in array -%}`
			`{{ item }}{%- br -%}`
			`{%- endfor -%}`
			`EndSplit`
			Output
			`Split content:`
			` A`
			` bcd`
			`EFG`
			`%`
			`++`
			`- /`
			`3432`
			`EndSplit`

splitByCharacterTypeCamelCase(String str)	Splits a String into an array of String with each element containing a substring based on the character type with the special case to maintain one upper case letter followed by some lower case letters as an element.
	Character types include: lower case, upper case, numbers, whitespace, math operators
		String[]	**Example**
			`{% set array = StringUtils.splitByCharacterTypeCamelCase(" AbcdEFGhij%++-/3432 ") %}`
			`Split content:{%- br -%}`
			`{%- for item in array -`

© Copyright, WhereScape Software Ltd All rights reserved.

Function	Description	Datatype	
			%} {{ item }}{%- br -%} {%- endfor -%} EndSplit **Output** Split content: Abcd EF Ghij % ++ - / 3432 EndSplit
swapCase(String str)	Swaps the case of a String changing upper and title case to lower case, and lower case to upper case.	String	**Example** {{ StringUtils.swapCase("UPPER lower") }} **Output** upper LOWER
trim(String str)	Removes whitespace from both ends of the String.	String	**Example** T{{ StringUtils.trim(" island ") }}T **Output** TislandT
uncapitalize(String str)	Uncapitalizes a String, changing the first character to lower case .	String	**Example** {{ StringUtils.uncapitalize("WORD") }} **Output** wORD

© Copyright, WhereScape Software Ltd All rights reserved.

Template Usage

If a template exists of the correct **Type** and **Target DB**, templates can be specified and evaluated as follows:

Operation	Location to Specify the Template	Notes
Alter Table DDL	**Connection Properties > Target Settings > Default Table Alter DDL Template**	Only for Custom database targets (on SQL Server, Oracle and Teradata).
Create DDL	**Table Properties > Storage > Create DDL Template** **Connection Properties > Target Settings > Default Table Create DDL Template** **Connection Properties > Target Settings > Default View Create DDL Template**	DDL is evaluated for the object at runtime, if a template is specified in the **Storage** tab. Alternatively, clicking the **Derive DDL** button in the **Override Create DDL** tab generates the Override DDL based on the specified template. **Important** If an Override DDL is specified, the Override DDL is used at runtime.
Create Index DDL	**Connection Properties > Target Settings > Default Index Create DDL Template**	Only for Custom database targets (on SQL Server and Oracle).
Drop Index DDL	**Connection Properties > Target Settings > Default Index Drop DDL Template**	Only for Custom database targets (on SQL Server and Oracle).
Export Script	**Export Object Properties > Script Template**	
Load Script	**Load Table Properties > Script Template** **Connection Properties > New Table Default Load Script Template**	
Update Procedure	When building the Update Procedure, specify a template to use in the **Template** drop-down field of the **Update Build Option > Processing tab**.	• If a **Block** template is specified, an SQL block is generated. • If a **Procedure** template is specified, an update procedure is generated.
Custom Update Procedure	**Table Properties > Custom Procedure**	Only on SQL Server, Oracle and Teradata
Update Script	**Table Properties > Update Script**	Only for Custom database targets (on SQL Server, Oracle and Teradata).
Post Load Procedure	**Load Table Properties > Post Load Procedure**	Only on SQL Server, Oracle and Teradata.
Post Load Script	**Load Table Properties > Post Load Script**	Only for Custom database targets (on SQL Server, Oracle and Teradata).

To check which Operations are supported by Templates on your Target database, refer to **Templates** for details.

Notes:

- The template of the right type and database controls the availability of the template options above, e.g. the template selection fields are only available if valid templates exists.
- Most of the default settings for Templates are configurable in the **Connection Properties > Target**

© Copyright, WhereScape Software Ltd All rights reserved.

Notes:

Setting tab. Refer to **Connection Target Settings** for details.

- The template object context menu that is displayed when you right click a Template object from the Objects list on the left pane provides a Usage Report option, which enables you to list the table objects that are currently associated with the selected template. Refer to **Working with Objects > Templates** for details.
- Additional host script language types which can be applied to templates can be defined in RED. Refer to **Host Script Languages** for details.

Script Templates for Custom Database Table Objects

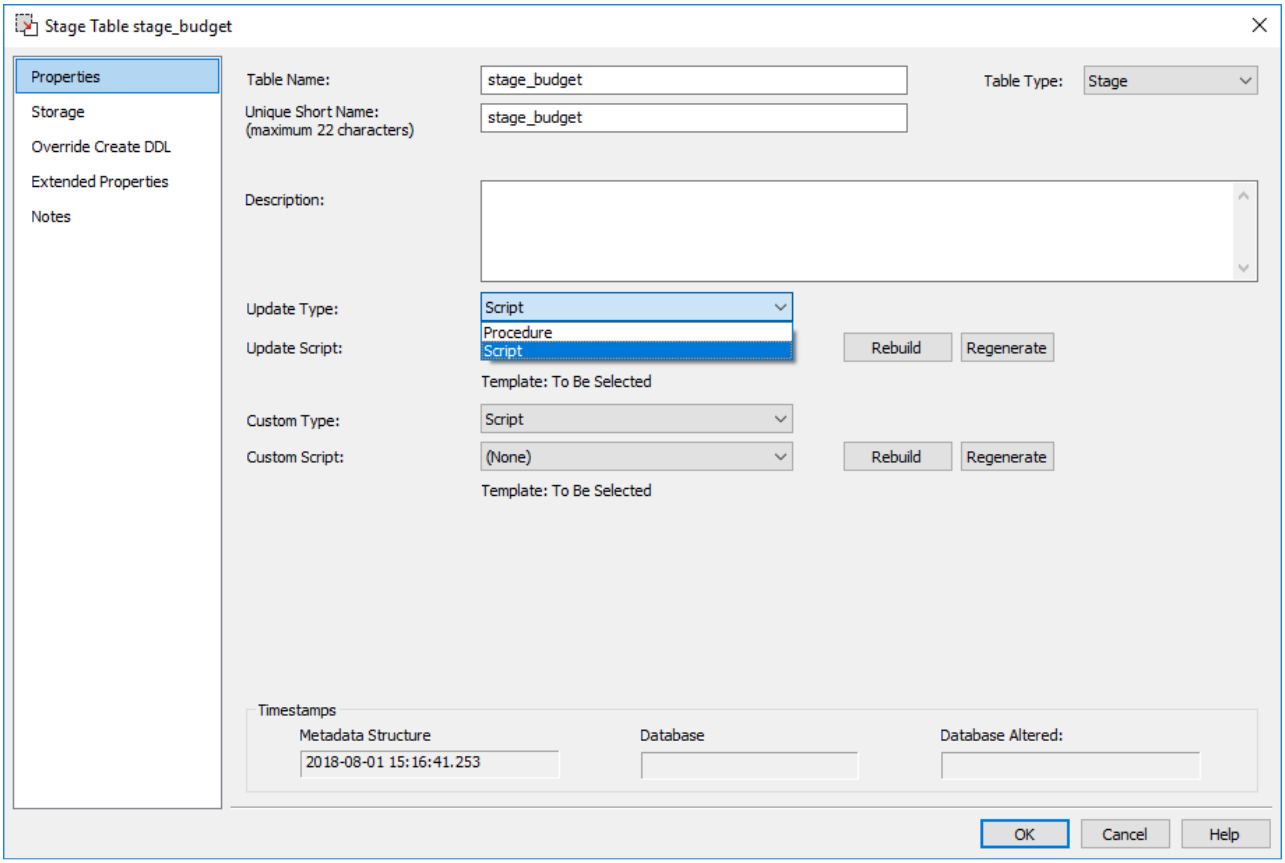
The update process for data warehouse table objects stored in a custom database target can be generated via a SQL block or a Windows PowerShell script. In both cases, templates are utilized to build the update procedures to populate the tables.

WhereScape RED provides PowerShell script templates that are designed to work for Table objects with a custom database target in a SQL Server or Oracle metadata repository. Refer to the custom database enablement packs provided by WhereScape for details about these PowerShell script templates. Moreover, additional host script language types that can be applied to Templates can be defined in RED via the **Tools > Host Script Languages** tool. User defined script templates can also be used for building the update procedures of Table objects stored in a custom database target.

The **Properties** window for table objects that support script based update has an **Update Type** drop-down field, which enables you to select between **Procedure** or **Script** based type processing for the table's update procedures.

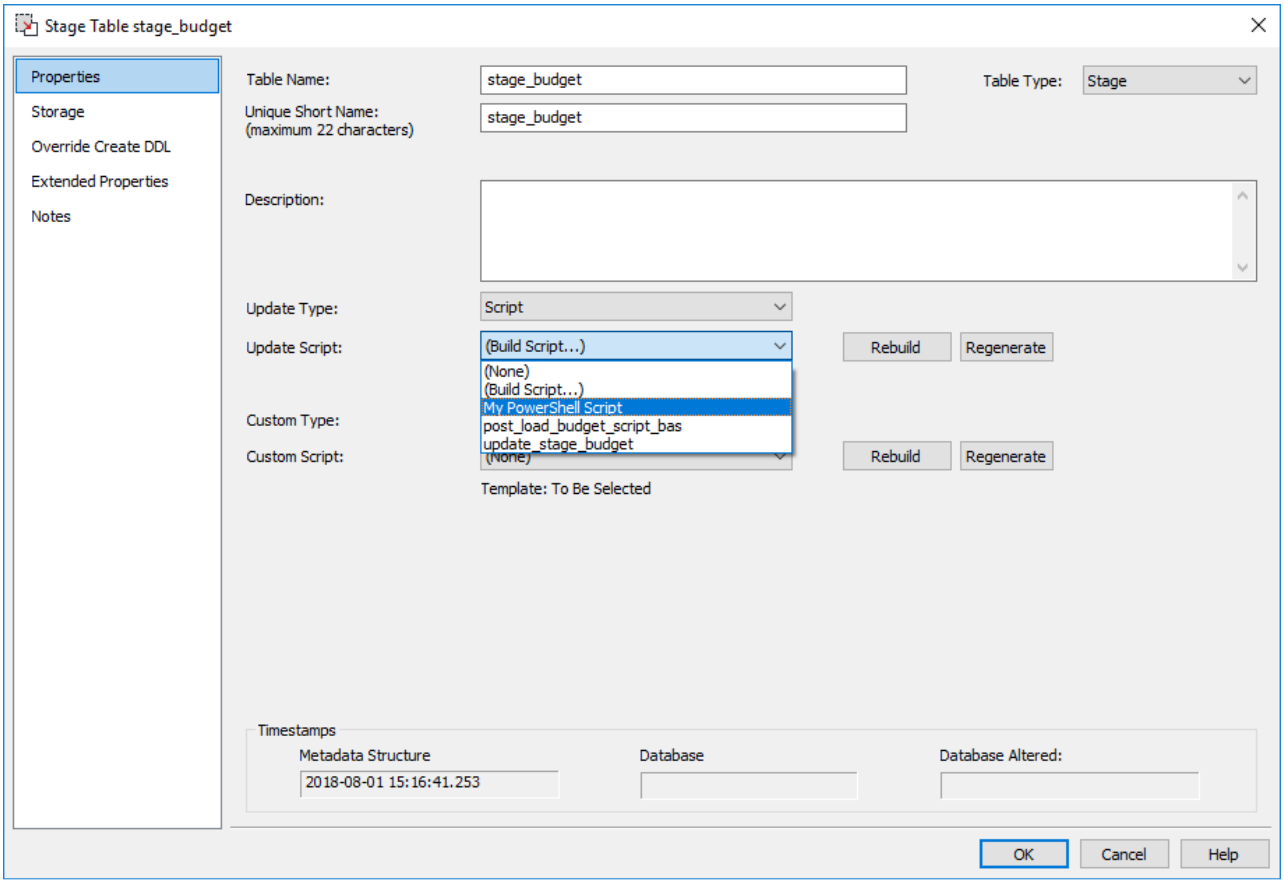
Note:

For new data warehouse table objects in a custom database target, the default **Update Type** is set to **Script**.



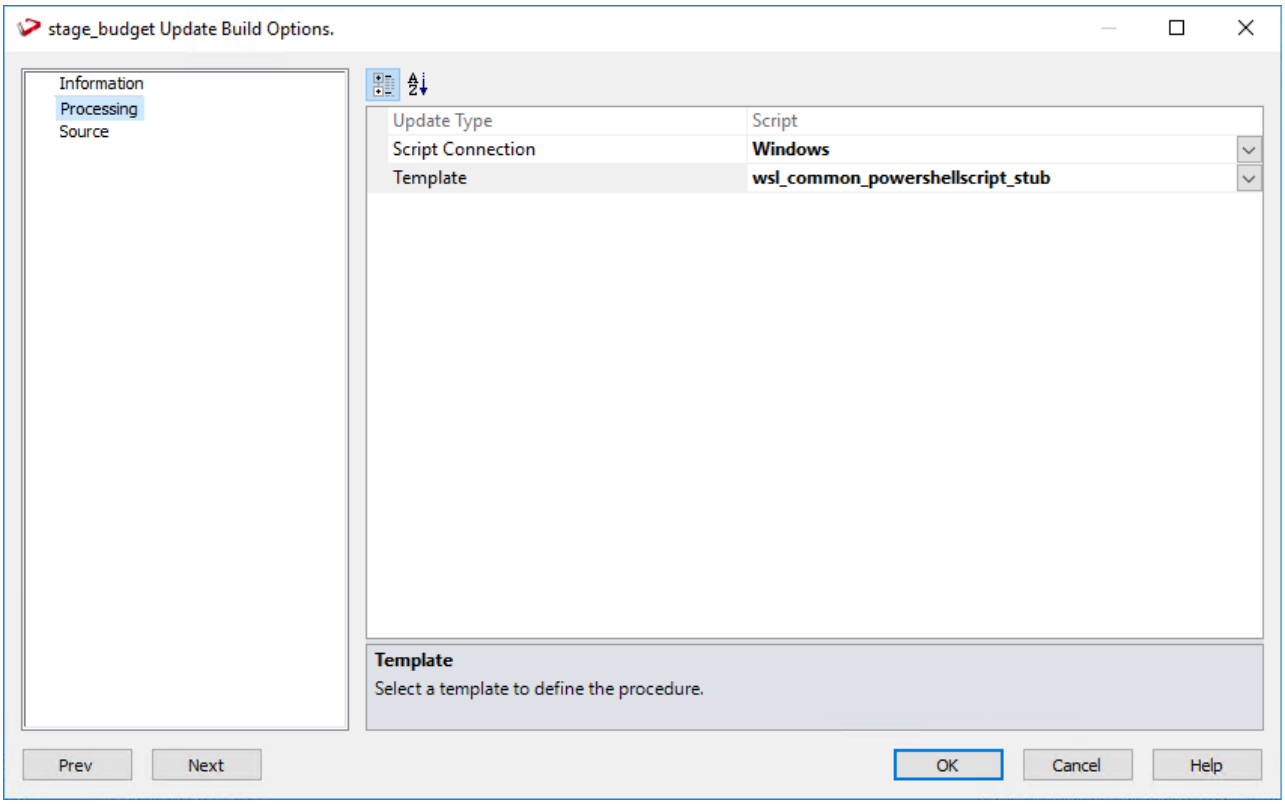
© Copyright, WhereScape Software Ltd All rights reserved.

Selecting **Script** from the **Update Type** drop-down displays the **Update Script** drop-down which enables you to select a script to use from the list of available script templates. The same applies to the **Custom Type** drop-down field.



Selecting the **(Build Script...)** option and then clicking the **Rebuild** or **OK** button opens the table **Update Build Options** window which enables you to define the **Script Connection** and **Template** to use. The same applies to the **Custom Script** processing.

© Copyright, WhereScape Software Ltd All rights reserved.

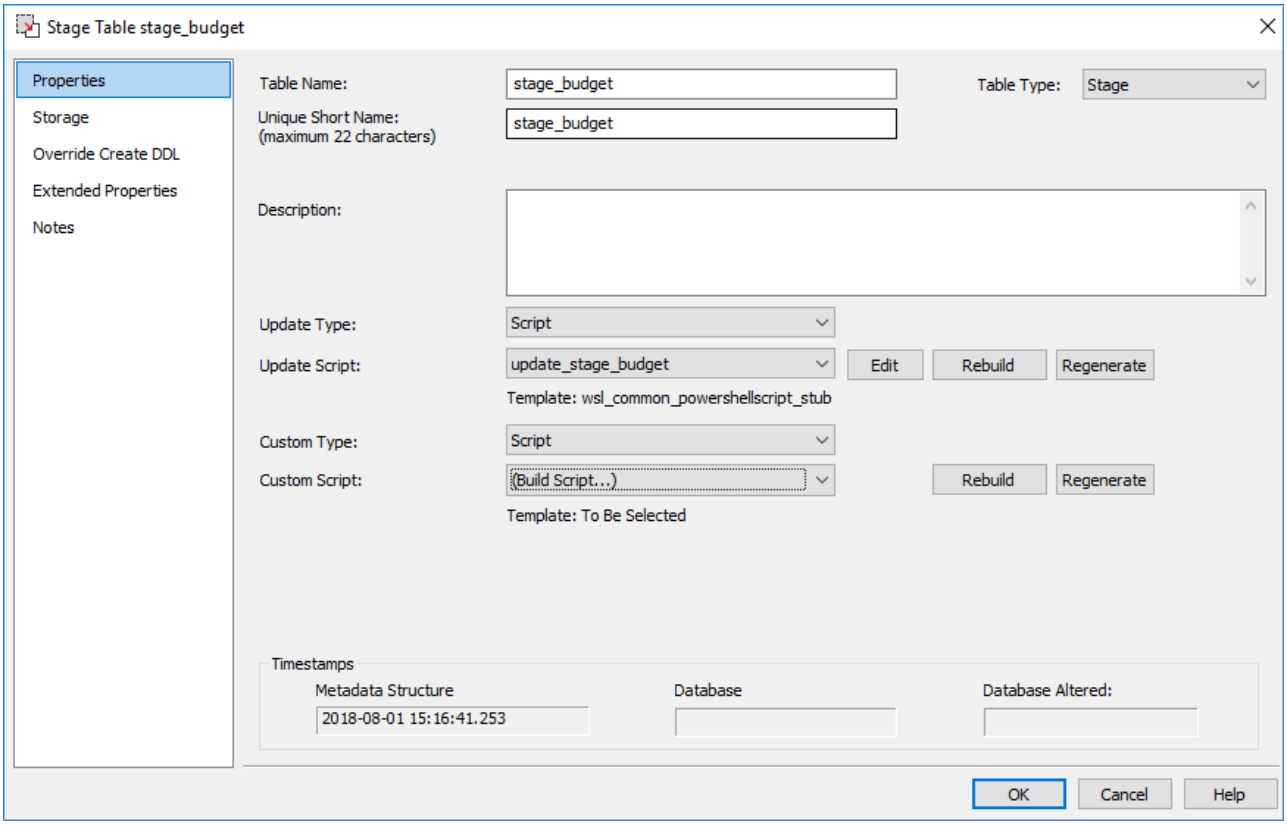


- Clicking the **Edit** button opens the selected update script template in the WhereScape RED **Script Editor** which enables you to edit the contents as required.
- Clicking the **Regenerate** button regenerates the selected update script template without any prompts, unless you made changes that require further input.
  The same applies to the **Custom Script** processing.

Note:

If a script (e.g. update script, custom script, load script, export script, etc.) is modified, the update time for that script is reflected in the **Script Editor** and the ### MODIFIED ### notice is displayed in the table object's Properties window when an update script is selected.

© Copyright, WhereScape Software Ltd All rights reserved.



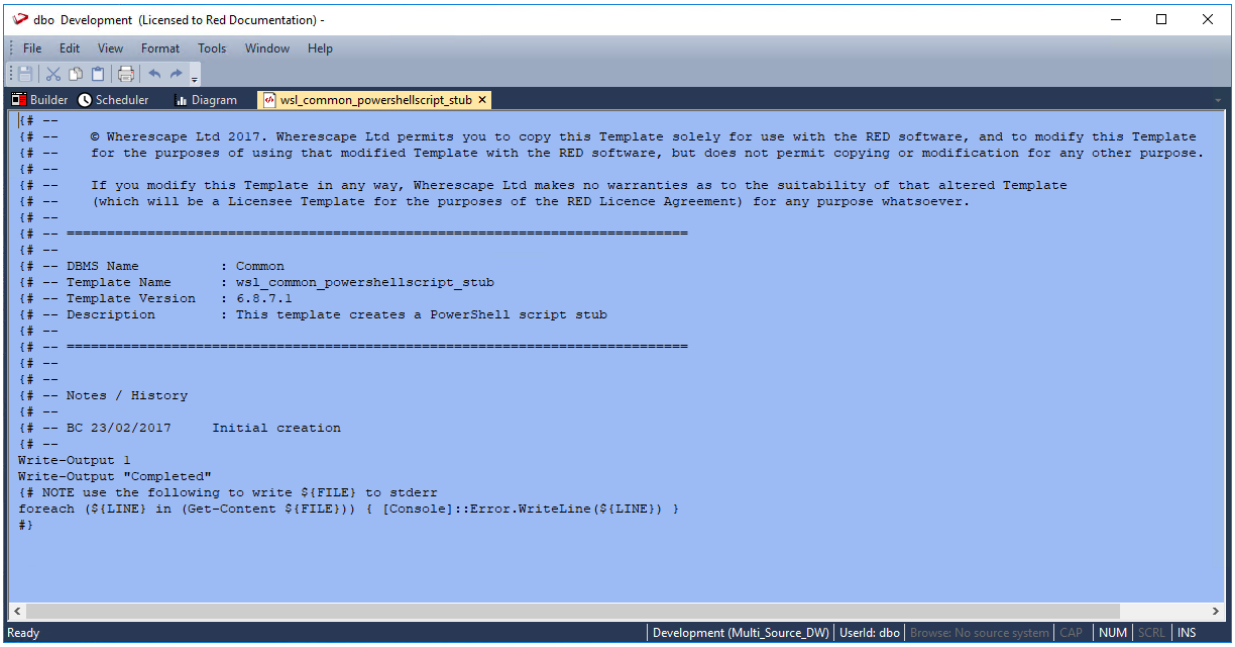
© Copyright, WhereScape Software Ltd All rights reserved.

Windows PowerShell Templates

PowerShell Stub Template (wsl_common_powershellscript_stub)

You can use the basic PowerShell stub template available in WhereScape RED that serves as a guide in using a template to generate a PowerShell script.

Additional PowerShell Templates can be downloaded from the WhereScape website (https://www.wherescape.com/support/software-downloads-documentation/templates/).

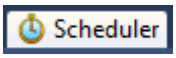


Notes:

- The PowerShell stub template (wsl_common_powershellscript_stub) is included under the **Template** objects list pane, along with the other templates available in WhereScape RED. If this stub template is not visible after installing/upgrading WhereScape RED, use the WhereScape RED Setup Administrator to Validate the Metadata Repository. For more information, please refer to the WhereScape RED Installation Guide.
- WhereScape provides a Windows application extension called WslMetadataServiceClient that can be used from PowerShell templates to simplify access to RED metadata content. This DDL is accessed from WSL provided templates in custom database enablement packs.
  Please contact your WhereScape customer representative, if you require details on how to use this DDL.

© Copyright, WhereScape Software Ltd All rights reserved.

Scheduler

The scheduler is accessible by clicking the **Scheduler** button 

from the toolbar. It is also available as a stand alone utility. In this way, operators can be given access to the scheduler without gaining full access to the data warehouse.

The scheduler runs on either a UNIX host or under Windows (as a system service). It processes predefined jobs, recording the success, failure or otherwise of the job.

Audit trail

Specific information relating to the tasks in the job are recorded to the audit trail. Generally, only summary information is written to this audit trail. The contents of the audit trail are maintained even after a job is deleted.

Error trail

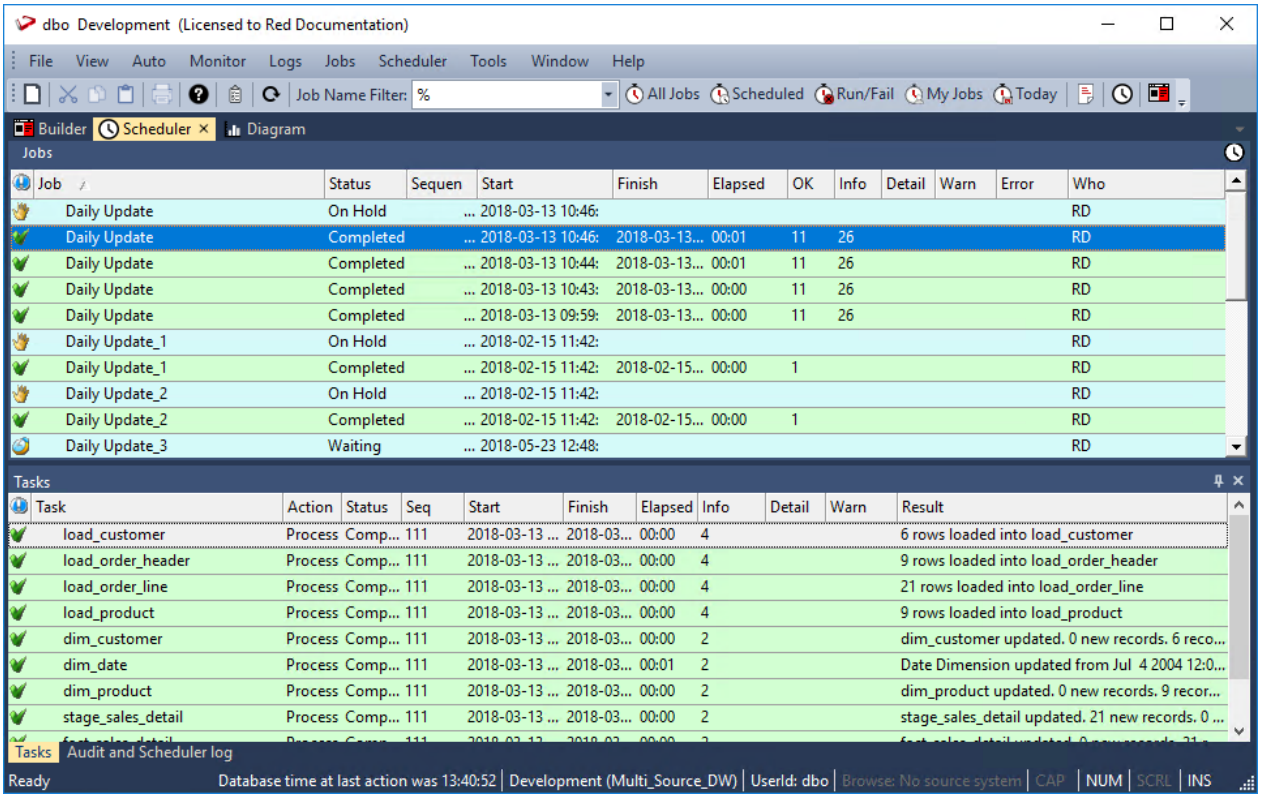
Detail or error information is written to the error trail. The contents of the error trail are deleted when the job is deleted.

Administration Views

It is possible to view the status of a job without using the WhereScape RED product. Three views are provided to assist in this undertaking. They are **ws_admin_v_audit**, **ws_admin_v_error** and **ws_admin_v_sched**. Queries can be issued using these views to see the results or status of a job.

Scheduler Window

An example of the **Scheduler** screen is shown below.



Toolbar/Jobs menu

Quick access to some job categories are in the toolbar. The complete options are listed under the **Jobs** menu and while most are self-explanatory they are described below:

Object	Description
Job Name Filter	Lists the jobs whose names match the filter supplied. This filter only works as an

© Copyright, WhereScape Software Ltd All rights reserved.

Object	Description
	appended filter to the main filter selected under **Jobs**, e.g. first enter a filter for **Job Name Filter** or select a filter from the drop-down list; and then choose your main filter under **Jobs**—e.g. All Jobs, Scheduled Jobs, etc.
All jobs	Lists all jobs in the middle pane
Scheduled	Lists the jobs that are waiting to run or are on hold.
Run/Fail	Lists the jobs that are running and those that have failed.
My Jobs	Lists the jobs you have scheduled or have run.
Today	Lists the jobs you have scheduled for today.
Last 24 hours	Lists the jobs that have run or started to run during the last 24 hours.
Prior 24 hours	Lists the jobs that ran during the previous 24 hours.
This Weeks Jobs	Lists the jobs that have run or are scheduled to run during the current week
Last Weeks Jobs	Lists the jobs that ran during the last week.
Recent log entries	Displays the Audit and Scheduler log which details the audit trail information of a task/job.
Scheduler status	Lists all schedulers and displays their current status. The status is updated every few minutes, and a right menu option enables the polling of a scheduler for status, and the termination of a scheduler.

Top pane

The top pane shows the details of the jobs. Information covers:

Column	Description
Job name	The name given to the job when created.
Status	The status of the job. Refer to the following section for the various status values.
Sequence	This is a unique number assigned to each job iteration and job. If you enter a new job, it acquires a new sequence number. In normal daily processing when no new jobs have been created, sequence numbers will be sequential.
Start and Finish times	The start and finish dates and times for the job.
Elapsed time	The time that elapses from start to finish of a job.
Ok	These are success messages written to the audit trail.
Info	Informational messages written to the audit trail about the running of the job.
Detail	Lines written to the detail or error logs.
Warn	The number of warnings written to the audit trail.
Error	The number of error messages written to the audit trail.
Who	The initials of the person who scheduled the job.

Additional fields can be added via the **Tools > Select Job Report Fields** option.

Middle pane

The middle pane shows the tasks related to a selected job. Task information available includes:

Column	Description
Task	The object name.
Action	The action to be done to the object.
Status	Status of the task.
Sequence	This is a unique number assigned to each job iteration and job. If you enter a new job, it acquires a new sequence number. In normal daily processing when no new jobs have

© Copyright, WhereScape Software Ltd All rights reserved.

Column	Description
	been created, sequence numbers will be sequential.
Start and Finish times	As the names suggest, the start and finish dates and times for the task
Elapsed time	The time that elapses from start to finish of a task.
Info	Informational messages written to the audit trail about the running of the job.
Detail	Lines written to the detail or error logs.
Warning	The number of warnings written to the audit trail.
Result	Result of the task.

Additional fields can be added via the **Tools > Select Task Report Fields** option

Bottom pane

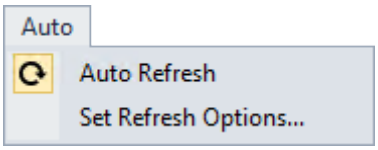
The bottom pane shows the audit trail of a selected task/job. Audit trail information includes:

Column	Description
Task	The object name.
Status	The status of the message.
Sequence	This is a unique number assigned to each job iteration and job. If you enter a new job, it acquires a new sequence number. In normal daily processing when no new jobs have been created, sequence numbers will be sequential.
Timestamp	Time of message output.
Message	The message.
DB Message	Message from database.
Job	Job relating to this message.

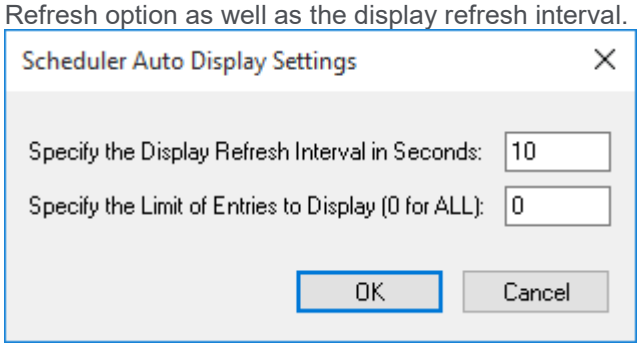
© Copyright, WhereScape Software Ltd All rights reserved.

Auto Menu

The **Auto** menu provides options for refreshing the scheduler display.



The following options are available:

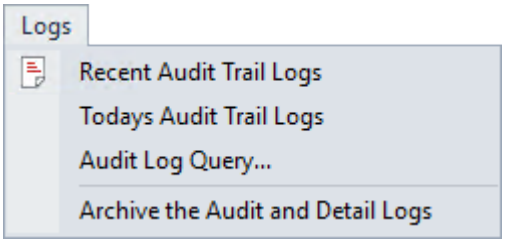
Option	Description
Auto Refresh	Use auto refresh to automatically refresh the display status of all jobs.
Set Refresh Options	Displays the Scheduler Auto Display Settings window which enables you to specify auto display settings. Use this setting to control the maximum number of rows that are displayed via the Auto Refresh option as well as the display refresh interval.  When clicking the **Set Refresh Options**, the settings dialog allows adding a display limit. The jobs displayed when on auto refresh will stop at the selected number of rows, so if the limit is set to 100, the refresh will stop after the first 100 rows (jobs) are returned. If **0** is selected in the Specify the Limit of Entries to Display, then all rows (jobs) is displayed.

Logs Menu

The **Logs** menu provides options for viewing the Audit Trail logs and enables you to archive logs when required.

The logs information is useful when a job fails to start or enters some other unknown state. Generally, the audit trail entries for a job can be found by drilling down into the job itself.

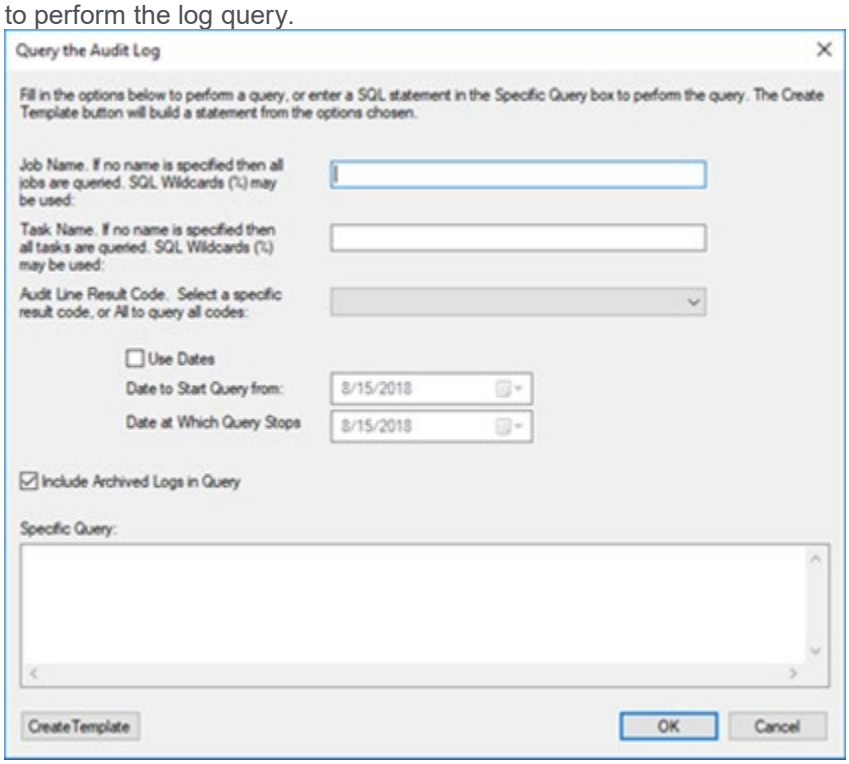
The Audit Trail log retains all the task information for a job, even if its associated job log is deleted. This log is backed up, as part of the metadata and should be periodically archived.



The following options are available:

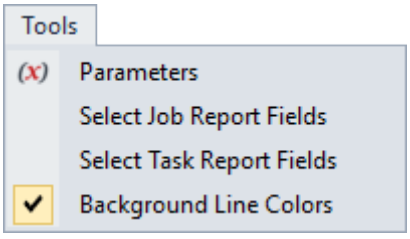
Option	Description
Recent Audit Trail Logs	Displays the Audit and Scheduler log which details the most recent audit trail information of a task/job.

© Copyright, WhereScape Software Ltd All rights reserved.

Option	Description
Todays Audit Trail Logs	Displays the logs for the jobs that have run in the current day.
Audit Log Query	Displays the **Query the Audit Log** window which enables you to specify the parameters to perform the log query. 
Archive the Audit and Detail Logs	Archives the Audit and Detail logs to an audit trail archive table. The archive process moves the records from the logs to an audit trail archive table. This table can be queried but is not backed up, as part of the metadata backup routines.

Tools Menu

The **Tools** menu provides options for configuring the scheduler display.



The following options are available:

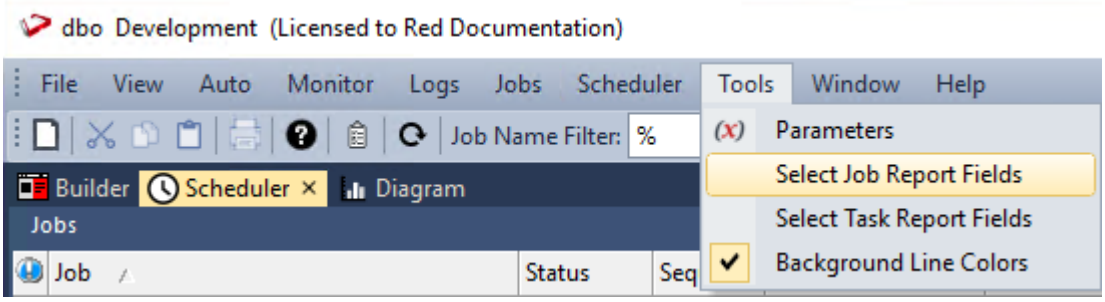
Option	Description
Parameters	Click to display all parameters defined in the meta data repository.
Select Job Report Fields	Click to choose the fields to include in the job report (see Select Job Report Fields below).
Select Task Report Fields	Click to choose the fields to include in the task report.
Background Line Color	Click to display/hide the background line colors in the Jobs and Tasks panes.

© Copyright, WhereScape Software Ltd All rights reserved.

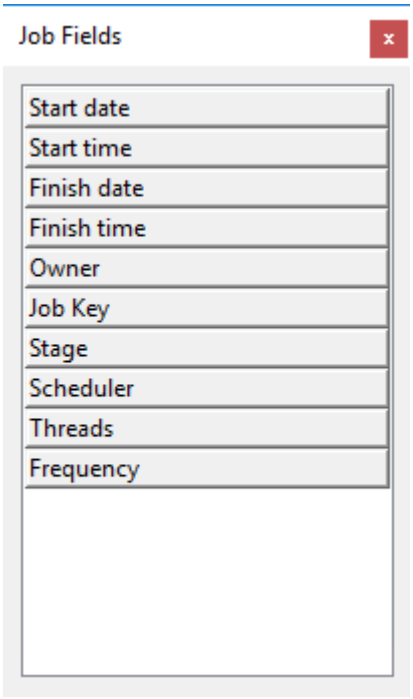
Select Job Report Fields

The **Select Job Report Fields** menu option in the Scheduler window enables you to select extra fields (e.g. Scheduler, Threads and Frequency) to include in the job report. such as fields into the job report.

1. To make these fields available, click **Tools > Select job report fields** from the top pane.



2. Select the fields you want to add to your job report from the menu and drag them to where you want to place them on the report.



Note

The "Frequency" field is only populated for scheduled jobs. If selected, it returns blank results for running or completed jobs.

© Copyright, WhereScape Software Ltd All rights reserved.

Scheduler States

A scheduled **job** can have the following states:

- Hold
- Waiting
- Blocked
- Pending
- Running
- Failed
- Failed - Aborted
- Completed

State	Description
Hold	The job is on hold. It can be edited and its state changed in order to release the job.
Waiting	The job is waiting to start, or waiting for its scheduled time to arrive, or waiting for a scheduler to become available.
Blocked	The job is blocked as a previous instance of the same job is still running. When the currently running instance completes this job will start.
Pending	This is the first stage of a running job. The scheduler has identified the job as ready to start and has allocated a thread, or sub task to process the job. A job is in this state until the thread or sub task begins processing. If a job stays in this state then the scheduler thread has failed for some reason. The logs can be checked on either the UNIX or Windows server on which the scheduler is running.
Running	The job is currently running. Double-click on the job name in the right pane to drill down into the specific tasks.
Failed	A failed job is one that had a problem. It can be restarted from the point of failure and is considered to be running unless subsequently aborted.
Failed - Aborted	The job has been aborted after having failed. Once in this state a job cannot be restarted. The job exists then only as a log of what occurred and is no longer regarded as a job.
Completed	The job has successfully completed, possibly with warnings. Once in this state a job cannot be restarted. The job exists then only as a log of what occurred and is no longer regarded as a job.

Note

When a job fails and drilling down does not show any errors against the tasks, right-click on the job and "**View Audit Trail**". The job may have failed because of an error in the Job level.

A scheduled **task** can have the following states:

- Waiting or Blank
- Held
- Running
- Failed
- Completed
- Error Completion
- Bad Return Status

State	Description
Held	The task has been held due to a prior dependency failure. The problem must be rectified and the job restarted.
Waiting (Blank)	Tasks that are waiting to run either due to a shortage of threads, or prior dependencies normally have a blank status.
Running	The task is currently running.

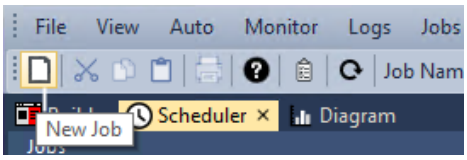
© Copyright, WhereScape Software Ltd All rights reserved.

State	Description
Failed	The task has had a fatal error. Any dependencies on this task will be held. Double-click the task to see more detail error information or review the audit and error/detail log for the job.
Completed	The task has completed successfully.
Error Completion	The task has completed with a handled Error. Any dependent tasks will be held, and the job must be restarted when the problem is rectified.
Bad Return Status	The task has returned an unknown status. This normally occurs with script files that produce unexpected information. The rule for scripts is that the first line returned must be a status of either 1, -1, -2, or -3. The second line is a message detailing the result. If the first line does not contain one of these four values, then this status will be returned and dependent tasks held. Run the script manually to view the output or check the logs.

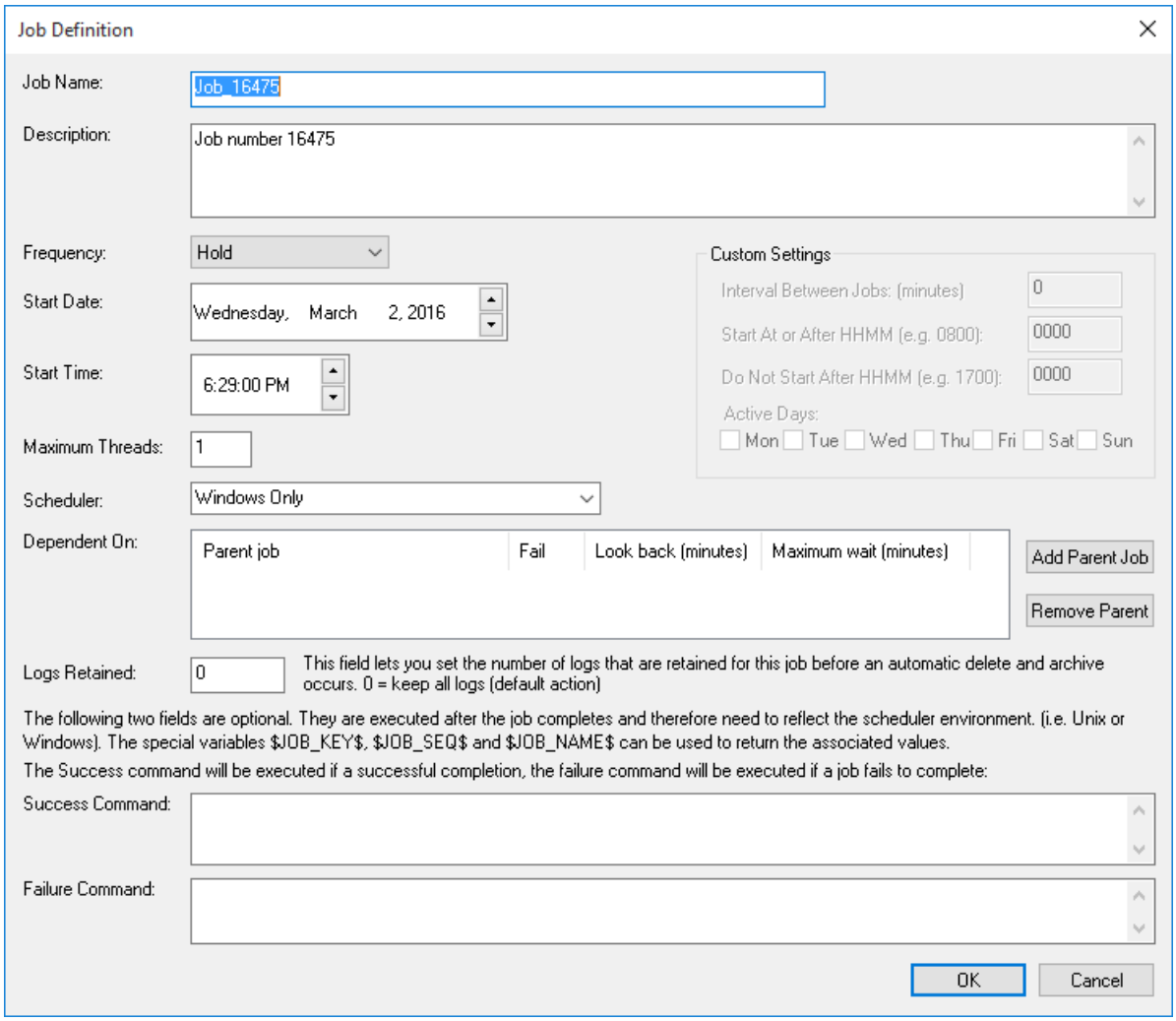
© Copyright, WhereScape Software Ltd All rights reserved.

Scheduling a Job

1. Open the Scheduler by clicking the **Scheduler** button from the toolbar.
2. Click **File > New Job** from the menu in the top of the screen, or click the **New Job** button from the toolbar.



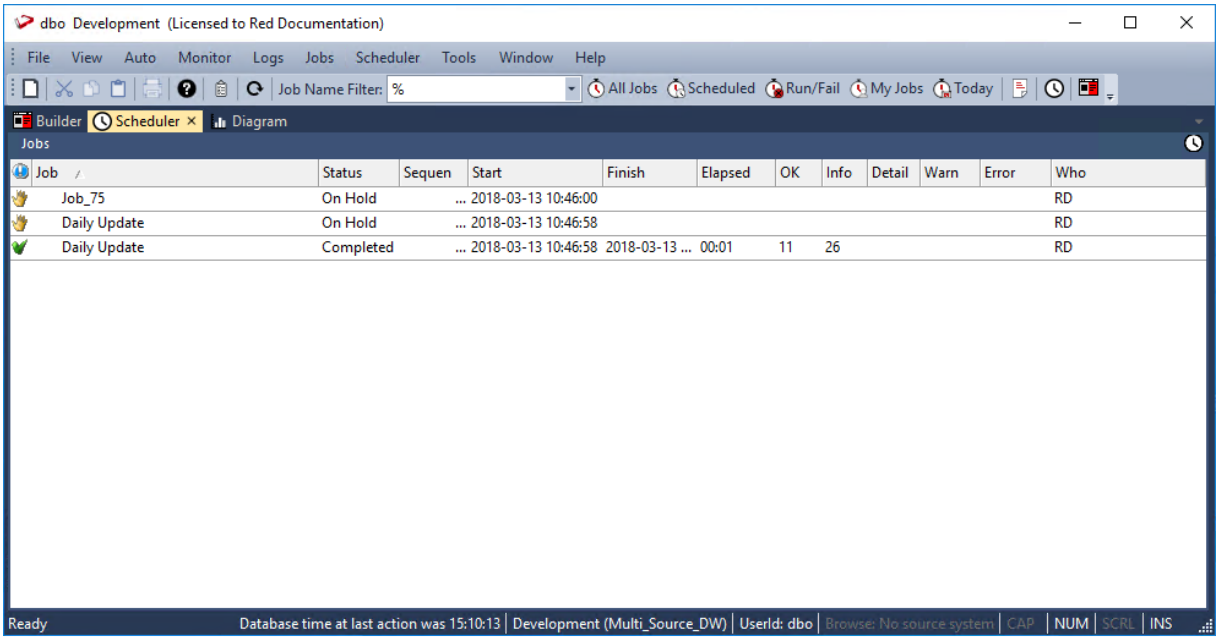
3. Enter the required details in the **Job Definition** window.



Refer to **Creating a Job** for more details on how to create a new job.

© Copyright, WhereScape Software Ltd All rights reserved.

4. Once the job has been created, click the **All Jobs** button from the toolbar. The newly created job is now displayed in the Scheduler window.



To create a job within a job

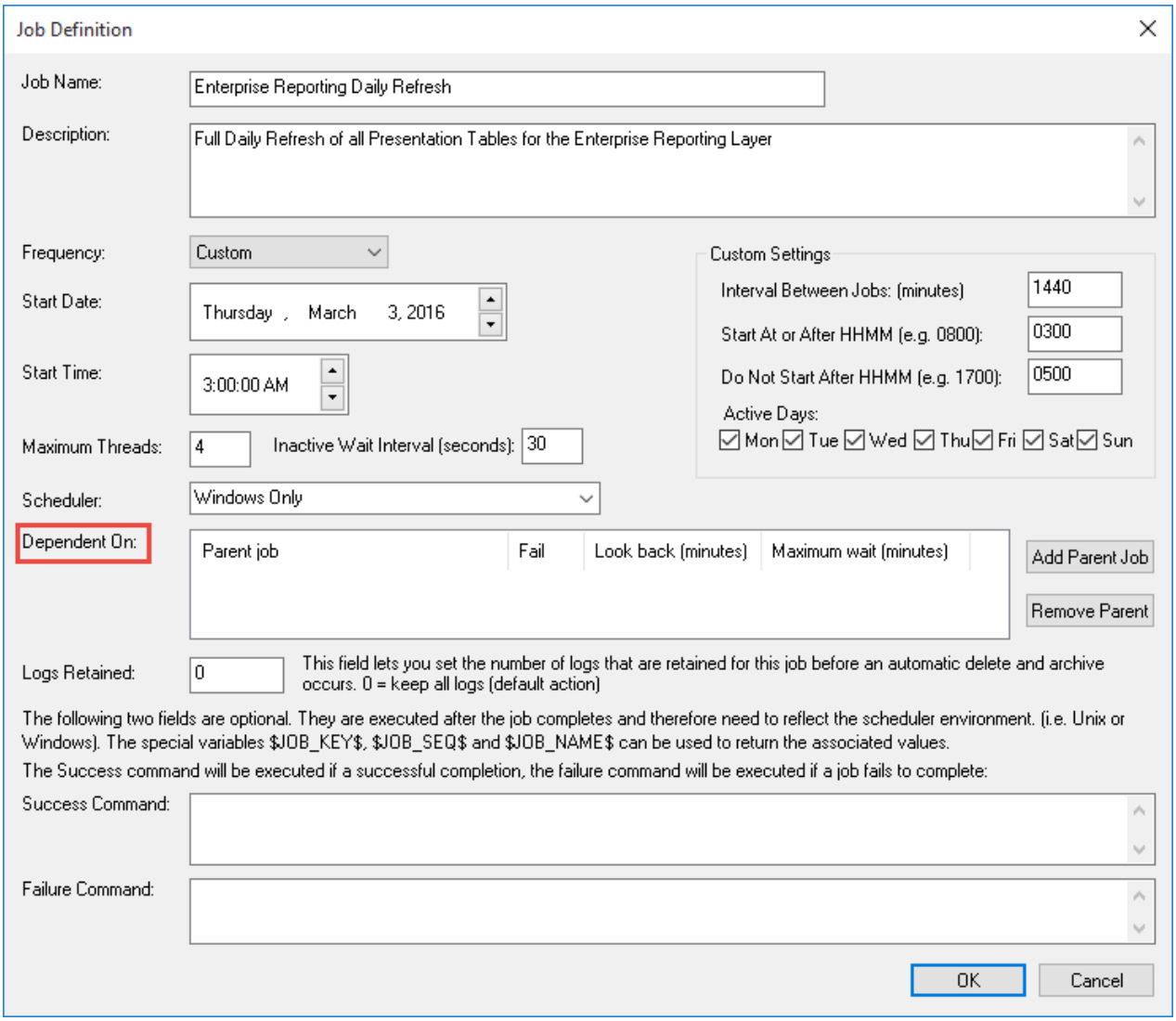
It is possible to schedule one job from another job. There are however some limitations and rules that must be understood when doing this.

1. A job that is called from another job is only ever allocated one thread. All tasks within the called job will therefore run sequentially.
2. A job can only have one running iteration. Therefore, a called job is blocked if that job is already running independently or as part of another job.
3. Any job dependencies for the called job are ignored. The parent's job dependencies are the only ones that are used.
4. A called job essentially runs as a separate job, so that if it fails both it and the parent job shows in a failed state. Once the problem is fixed, the parent job should be restarted which restarts the called job.

© Copyright, WhereScape Software Ltd All rights reserved.

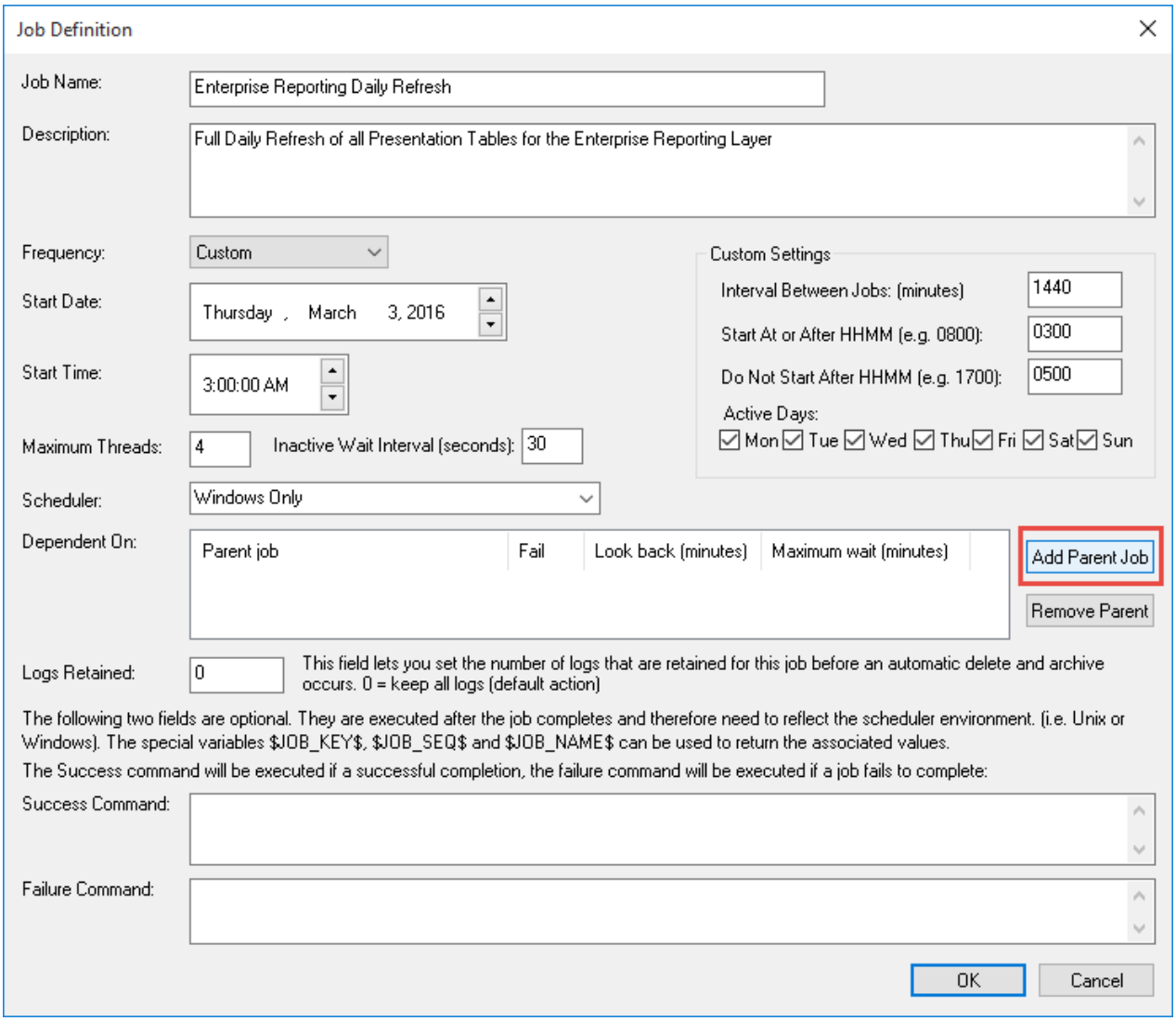
To create a job dependency

It is possible to make a job dependent on another job, using the **Dependent On** field in the **Job Definition** window.

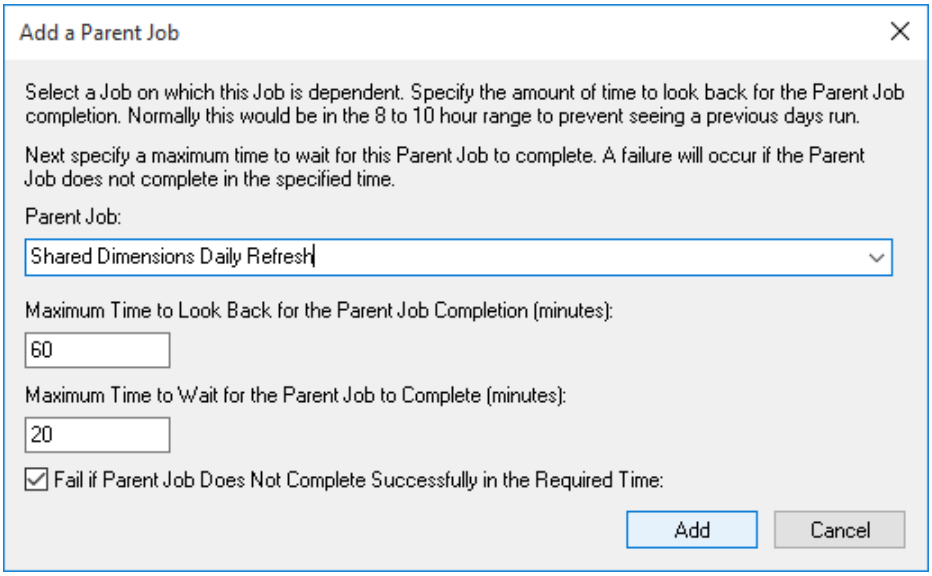


© Copyright, WhereScape Software Ltd All rights reserved.

Click the **Add Parent Job** button.



In the window that appears, select the **Parent Job** from the drop-down list. In our case we choose the job **Shared Dimensions Daily Refresh**.



The **Maximum Time to Look Back for the Parent Job Completion** field prevents older iterations of the parent job as being identified as a completion. In our example, we are starting both jobs at 3am, so we don't

© Copyright, WhereScape Software Ltd All rights reserved.

need to look too far back to ensure that the dimension refresh has run. We have therefore set the look back minutes to 60 to allow for any delays in starting this job.

The **Maximum Time to Wait for the Parent Job to Complete** specifies how long to await a successful completion of the parent job. In our example, we know that the dimension refresh only takes a few minutes, but we should allow for the occasional slow network or resource drains making the dimension refresh take longer; so we have set the maximum wait to 20 minutes. This means that our job will wait 20 minutes from its own scheduled start time for the parent job to complete.

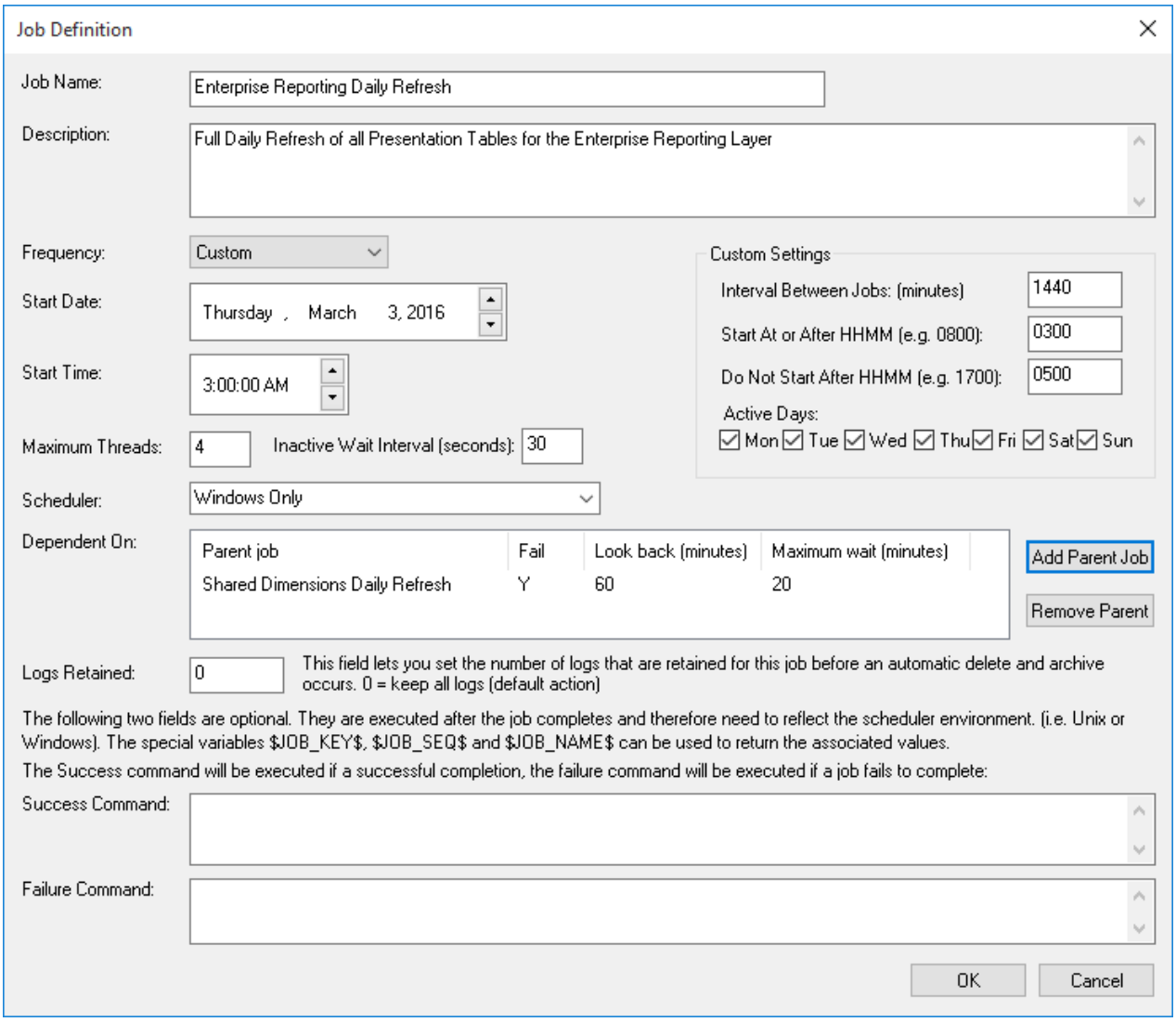
The check box to fail if the parent job does not complete in time prevents this job from running if the parent job (dimension refresh) does not complete successfully. We do not want the transactional data in our fact deliveries to be flagged with 'Unknown' dimensional item(s), so we can leave this check box selected to ensure that this job does not run.

Click **Add**.

Note:

Clearing the check box to fail if the parent fails will simply ensure that this job waits for the completion of the dimension refresh, and irrespective of the dimensions refresh success or failure, it starts.

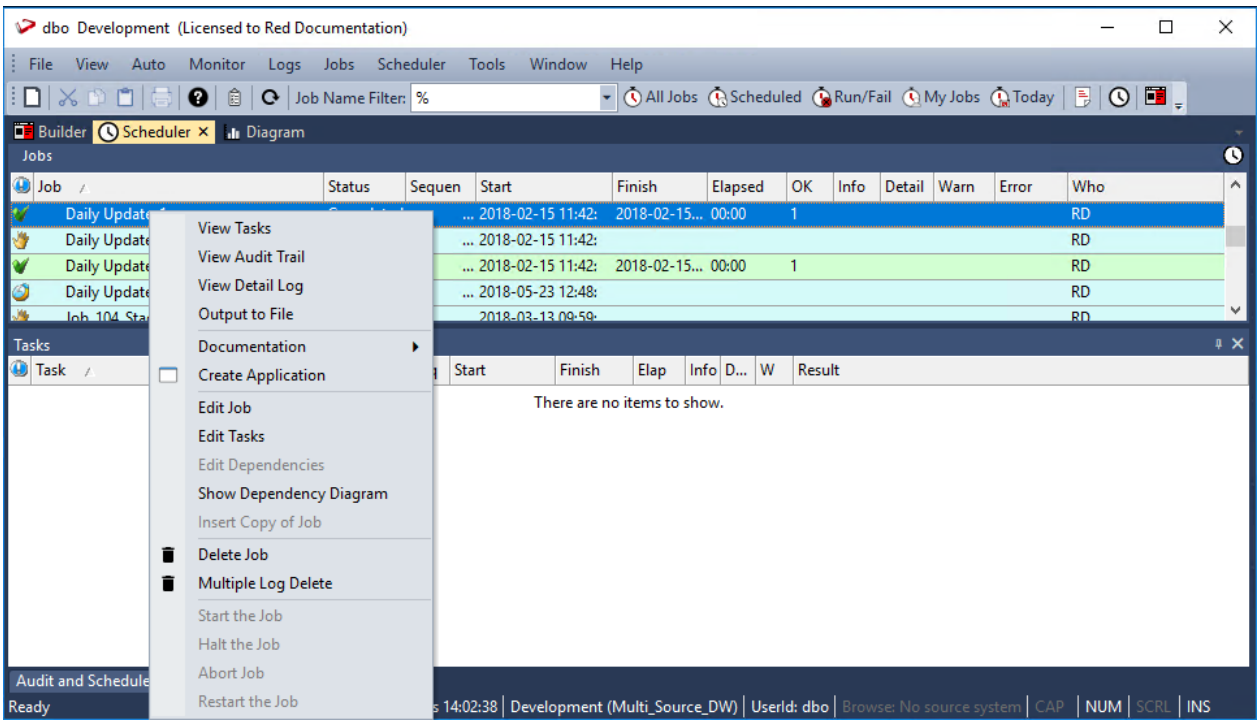
Click **OK** to link this data job to the parent dimensional job. In this way, the job **Enterprise Reporting Daily Refresh** cannot run until the parent job **Shared Dimensions Daily Refresh** has completed successfully; thus the facts will have the latest dimensional keys associated with them.



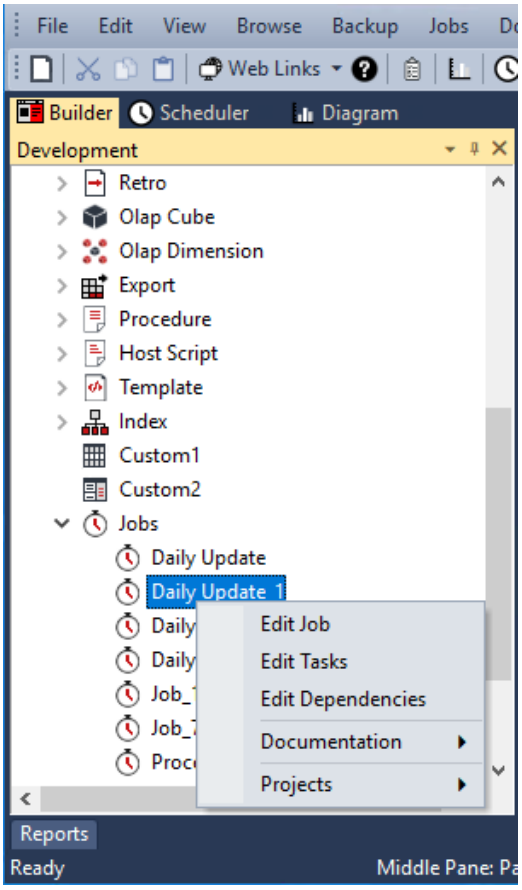
© Copyright, WhereScape Software Ltd All rights reserved.

Working with Jobs

When you right click a Job in the Scheduler window, the context menu provides several options for working with the job.



You can also access and manage the scheduler jobs from the Objects list pane of the **Builder** window.



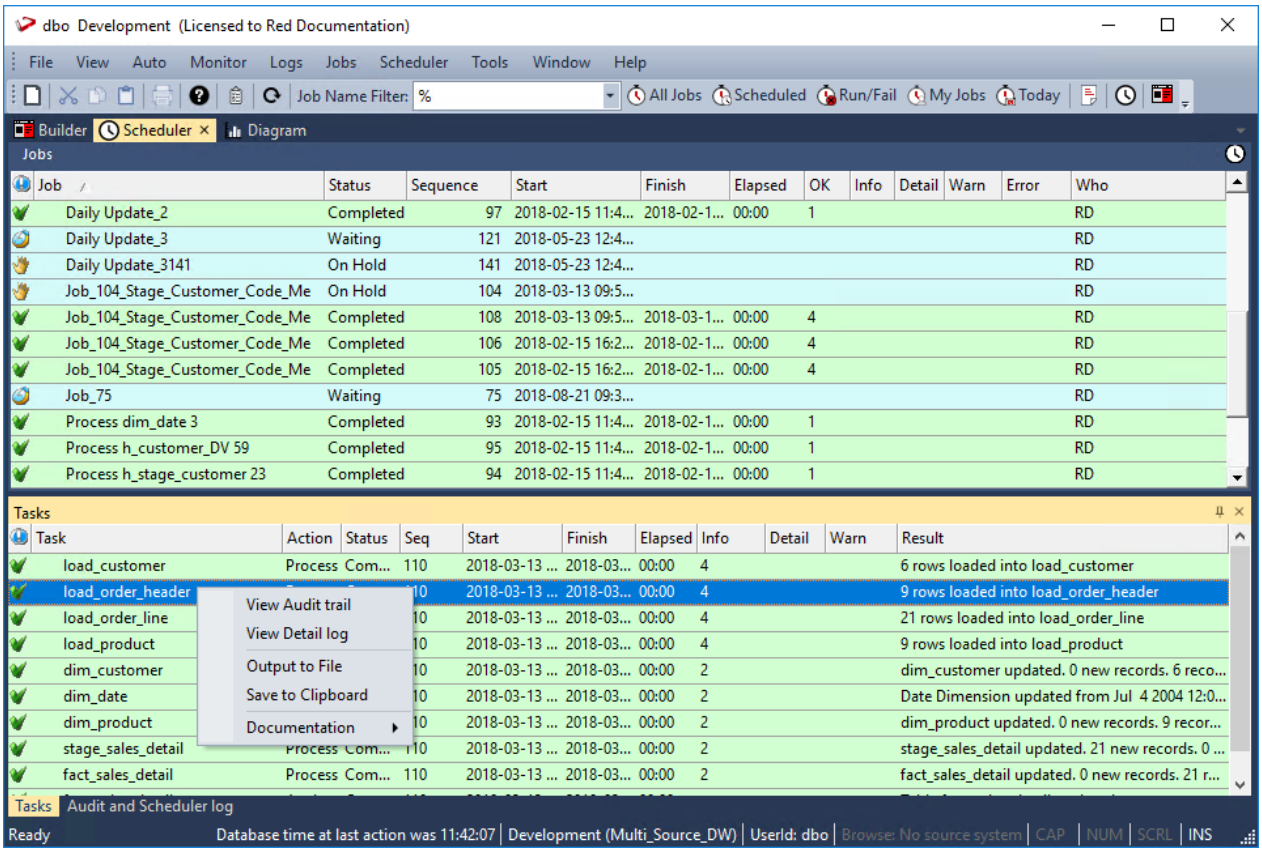
© Copyright, WhereScape Software Ltd All rights reserved.

Some of the options are discussed in more detail in the succeeding sections, however a brief overview of the menu options follows:

Option	Description
View Tasks	Enables you to view the tasks of a job.
View Audit Trail	Enables you to view the audit trail of a job.
View Detail Log	Enables you to view the detail log of a job.
Output to File	Enables you to export the jobs to a CSV file.
Documentation	Enables you to display or create documentation for a job.
Edit Job	Enables you to edit a job. Refer to **Editing a Job** for details.
Edit Tasks	Enables you to edit the tasks of a job. Refer to **Editing Tasks in a Job** for details.
Edit Dependencies	Enables you to edit the task dependencies of a job. Refer to **Editing Task Dependencies** for details.
Insert Copy of Job	Enables you to insert a copy of a job. Refer to **Inserting a Copy of a Job** for details.
Delete Job	Enables you to delete a job. Refer to **Deleting a Job** for details.
Multiple Log Delete	Enables you to delete multiple logs of a job. Refer to **Deleting Job Logs** for details.
Start the Job	Enables you to start a job. Refer to **Starting a Job** for details.
Halt the Job	Enables you to halt a job. Refer to **Halting a Job** for details.
Abort the Job	Enables you to abort a job. Refer to **Aborting a Job** for details.
Restart the Job	Enables you to restart a job. Refer to **Restarting a Job** for details.

Job Tasks

You can display the tasks associated with a job by right-clicking the job and selecting **View Tasks** from the context menu. The tasks are listed below the Jobs pane and provides the following context menu options:



© Copyright, WhereScape Software Ltd All rights reserved.

Option	Description
View Audit Trail	Enables you to view the audit trail of a task.
View Detail Log	Enables you to view the detail log of a task.
Output to File	Enables you to export the task details to a CSV file.
Save to Clipboard	• **Tasks** - Enables you to copy the contents of the **Result** column(s) of the selected line(s) from the **Tasks** list to the Windows clipboard. • **Audit and Scheduler log** - Enables you to copy the contents of the **Message** column(s) of the selected line(s) from the **Audit and Scheduler log** to the Windows clipboard.
Documentation	Enables you to display or create documentation for a task.

© Copyright, WhereScape Software Ltd All rights reserved.

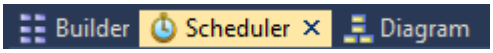
Creating a Job

Tip:

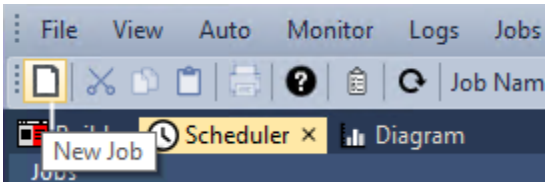
You can create a Scheduler job from:

- Projects or Project Groups—refer to **Building Scheduler Jobs from Projects or Groups** for details.
- Object Groups—refer to **Building Scheduler Jobs from Object Groups** for details.
- Diagrams—refer to **Creating a Job from a Diagram** for details.

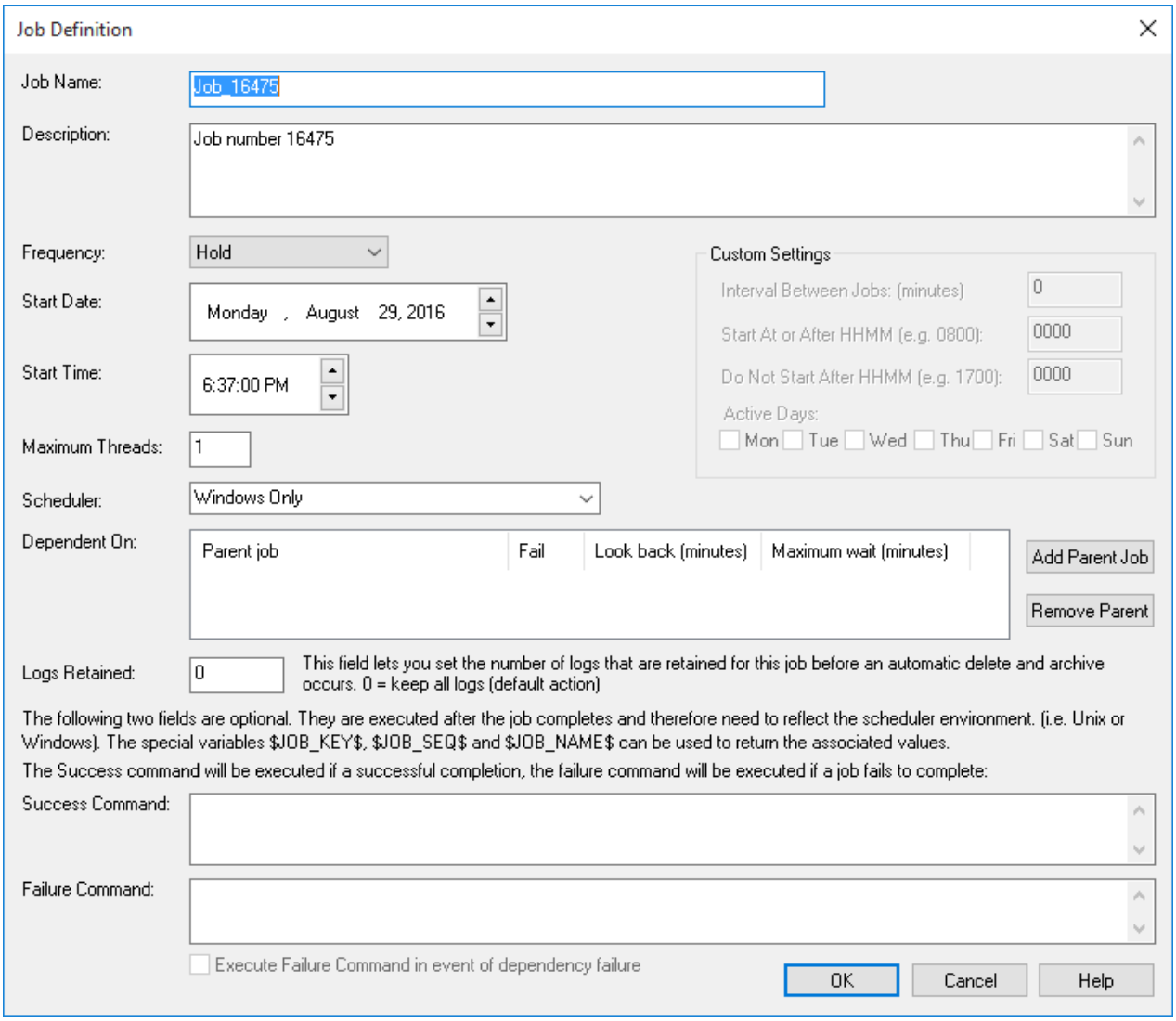
Click the **Scheduler** tab to open the **Scheduler** window.



Click the **New Job** button to create a new job.



A **Job Definition** window is displayed.



© Copyright, WhereScape Software Ltd All rights reserved.

Complete the fields and then click **OK**. The main fields are described in the following table:

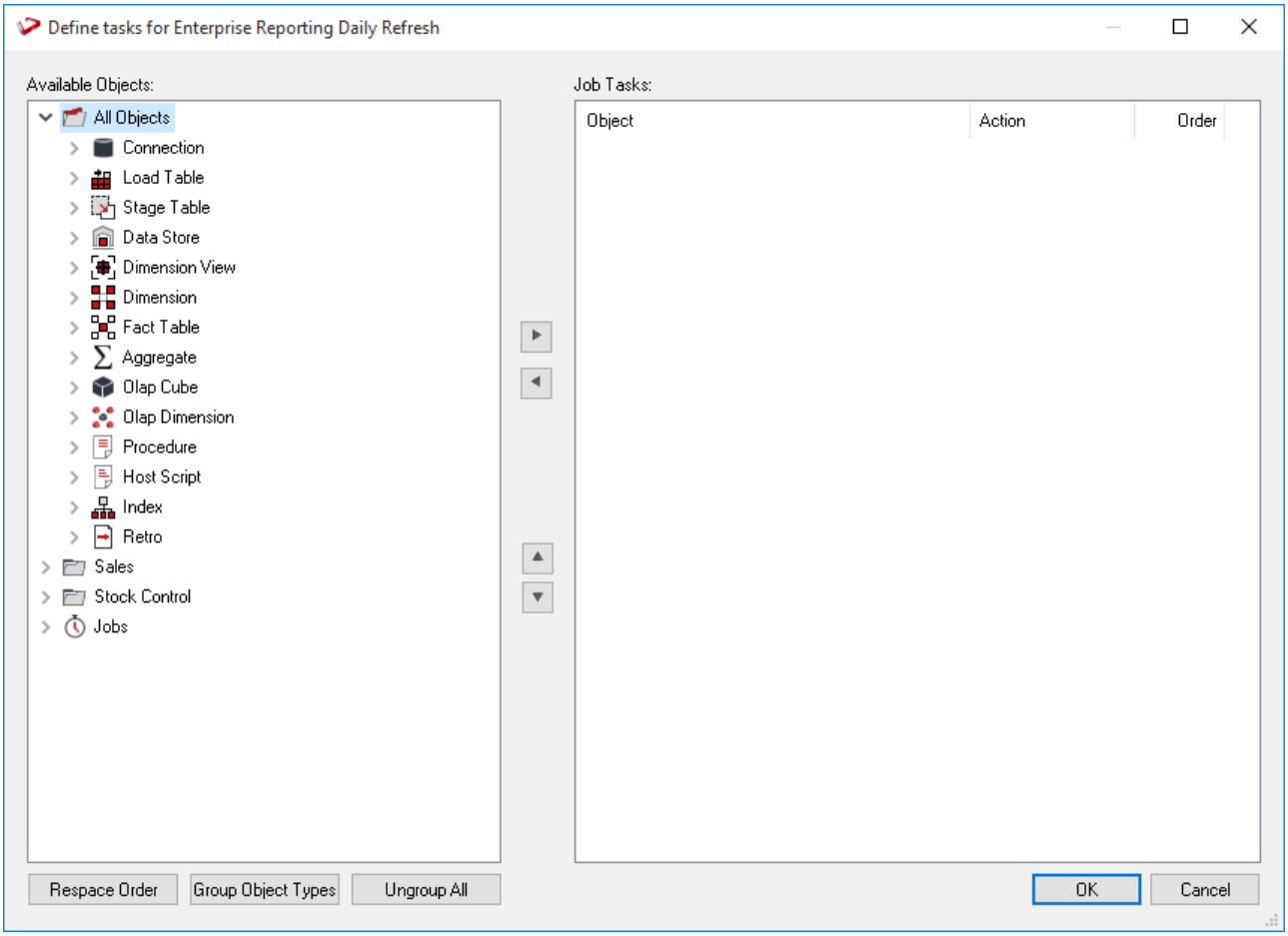
Field	Description
Job Name	The Scheduler defaults to the next job number in the sequence. You can alter this to any alphanumeric. **Tip** Only alphanumerics, spaces and the underscore are supported in the name. **Warning** On some UNIX systems, long job names can cause jobs to be canceled (see Knowledge Base article 67), so where possible keep the name short.
Description	A description of the job.
Frequency	When the job runs. The options available in the drop-down list box are: • Once Only - job is deleted on completion • Once and Hold - runs and puts another copy of the job on hold • Hold - puts the job on hold for manual release • Daily - runs the job daily • Custom - enables custom definition • Weekly - runs the job weekly • Monthly - runs the job monthly • Annually - runs the job annually
Start Date and Start Time	The date and time for the job to start.
Max Threads	The maximum number of threads allocated to run the job, e.g., if some tasks can run in parallel then if more than one thread is allocated then they will run in parallel.
Scheduler	Certain types of jobs will only run in a specific environment. For example ODBC based loads can only be handled by the Windows scheduler. It is possible to have multiple schedulers running. Select the desired scheduler from this drop-down. The valid options are: UNIX Preferred, UNIX Only, Windows Preferred, Windows Only, or the name of a specific scheduler can be entered (e.g. WIN0002)
Dependent On	A job can be dependent on the successful completion of one or more other jobs. Click the **Add Parent Job** button to select a job that this job will be dependent on. The maximum time to look back for parent job completion field prevents older iterations of the parent job as being identified as a completion. The maximum time to wait specifies how long to await a successful completion of the parent job. The action if that wait expires can also be set. Refer to the Job Dependency example in **Scheduling a Job** for details.
Logs Retained	Specify the number of logs to retain for the job. By default, all logs are retained. This field can be used to reduce the build up of scheduler logs by specifying a number of logs to retain.
Success command and Failure command	These are either UNIX or Windows shell commands depending on which scheduler is used. They are executed if the condition is met. Typically, these commands would mail or page on success or failure. **Notes:** • The RED scheduler does not check return codes from called commands, scripts and programs. • It is recommended that all output from commands, scripts and programs is re-directed to a log file. For example, add this to the end of any SUCCESS/FAILURE commands: >> c:\scheduler\success_failure_prod.log 2>&1

© Copyright, WhereScape Software Ltd All rights reserved.

The following fields are available if a frequency of **Custom** is chosen:

Field	Description
Interval between jobs (Minutes)	Specify the number of minutes between iterations of the job. For example, to run a job every 30 minutes set this value to 30. If a job is to run only once but on selected days set this value to 1440 (daily)
Start at or after HHMM	The time that the job may run from. To run anytime set to 0000.
Do not start after HHMM	If multiple iterations are being done then this is the time after which a new iteration will not be started. For example, if a job is running every 10 minutes it will continue until this time is reached. To run till the end of day set to 2400.
Active on the days	Select each day of the week that the custom job is to be active on.

Once the job itself has been defined, tasks then need to be added to the job. The **Define tasks** window is shown below.



The screen has two main areas. The right pane shows the tasks to be run for this job and the left pane lists all the objects.

Adding a task

Double-click an object in the left pane to add it to the task list in the right pane. Normally objects, such as Load or Fact tables are scheduled rather than procedures.

Setting the action on a task

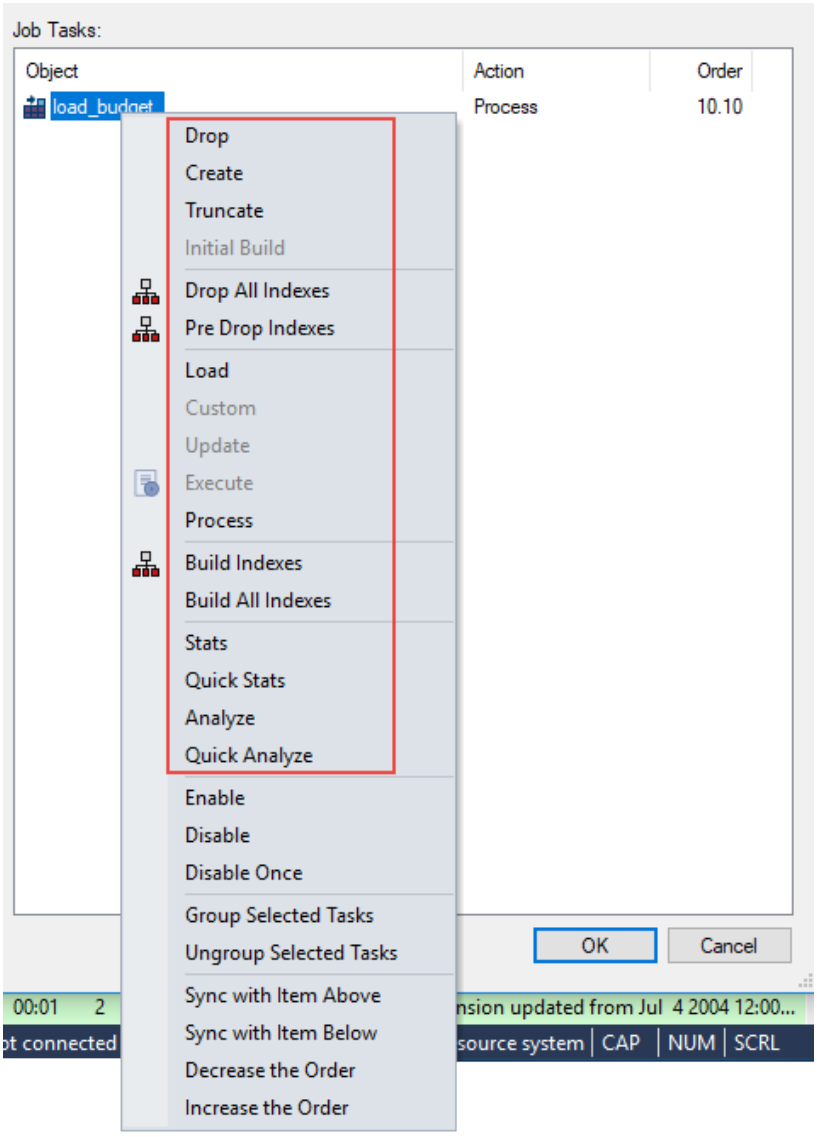
Each task can have a specific action that is to be performed on its object.

The default action for **load tables** is **process**. This means that when the task is actioned, it will drop any indexes that are due to be dropped, or have **pre-drop** set, then load the table and perform any post-load procedures or transformations and then re-create any dropped indexes.

© Copyright, WhereScape Software Ltd All rights reserved.

The default action for all other tables is the same as above, except it will execute the **update** procedure rather than loading the table.

You can change the action on a task by right-clicking the task in the right pane. The menu options are shown below.



The following task actions are available:

Action	Description
Drop	Drop table, view or index.
Create	Create table, view or index.
Truncate	Delete all rows from the table.
Initial Build	**Drop All Indexes** then **Custom** then **Build All Indexes**.
Drop All Indexes	Drop all indexes on the table.
Pre Drop Indexes	Drop all indexes on the table marked as "Pre Drop".
Load	Load the table (Load tables only).
Custom	Run the custom procedure on the table.
Update	Run the update procedure on the table.
Execute	Execute the procedure or host script.
Process	**Pre Drop Indexes** then **Update** and then **Build Indexes**.

© Copyright, WhereScape Software Ltd All rights reserved.

Action	Description
Process and Statistics	**Process** then Default Stats as defined in the Table Properties > Statistics > Process and statistics method (DB2 only).
Build Indexes	Build the indexes on the table marked as "Pre Drop".
Build All Indexes	Build all indexes on the table.
DB2: Analyze	Performs a custom analyze if defined in: • Table Properties > Statistics > Analyze else for tables performs: • **RUNSTATS ON TABLE tabname WITH DISTRIBUTION AND DETAILED INDEXES ALL** otherwise (for indexes): • **RUNSTATS ON TABLE indname FOR INDEXES**
DB2: Quick Analyze	Performs a custom analyze if defined in: • Table Properties > Statistics > Quick Analyze else for tables performs: • **RUNSTATS ON TABLE tabname** otherwise (for indexes): • **RUNSTATS ON TABLE indname FOR INDEXES**
DB2: Stats	Performs a custom stats if defined in: • Table Properties > Statistics > Stats else for tables performs: • **RUNSTATS ON TABLE tabname WITH DISTRIBUTION AND DETAILED INDEXES ALL** otherwise (for indexes): • **RUNSTATS ON TABLE indname FOR INDEXES**
DB2: Quick Stats	Performs a custom analyze if defined in: • Table Properties > Statistics > Quick Stats else for tables performs: • **RUNSTATS ON TABLE tabname** otherwise (for indexes): • **RUNSTATS ON TABLE indname FOR INDEXES**
Oracle: Analyze	Performs a custom analyze if defined in: • Tools > Options > Statistics > Table > Index Analyze Full otherwise performs: • **ANALYZE TABLE / INDEX name COMPUTE STATISTICS**
Oracle: Quick Analyze	Performs a custom analyze if defined in: • Tools > Options > Statistics > Table/ Index Analyze Quick otherwise performs: • **ANALYZE TABLE / INDEX name ESTIMATE STATISTICS SAMPLE 3 PERCENT**

© Copyright, WhereScape Software Ltd All rights reserved.

Action	Description
Oracle: Stats	Performs a custom stats if defined in: • Tools > Options > Statistics > Table/Index Stats Full otherwise performs: • **DBMS_STATS.GATHER_ TABLE / INDEX _STATS()** using the ownname and tabname / indname parameters with cascade.
Oracle: Quick Stats	Performs a custom analyze if defined in: • Tools > Options > Statistics > Table/Index Stats Quick otherwise performs: • **DBMS_STATS.GATHER_ TABLE / INDEX _STATS()** using the ownname and tabname/indname parameters with cascade and estimate_percent = 3.
SQL Server: Analyze	Performs: • **UPDATE STATISTICS name WITH FULLSCAN**
SQL Server: Quick Analyze	Performs: • **UPDATE STATISTICS name WITH SAMPLE 3 PERCENT**
SQL Server: Stats	Same as Analyze.
SQL Server: Quick Stats	Same as Quick Analyze.

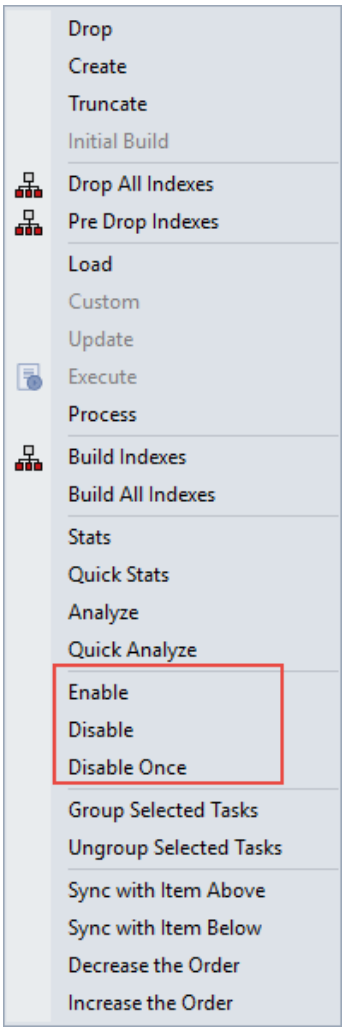
Note

Not all actions are available on all object types.

© Copyright, WhereScape Software Ltd All rights reserved.

Setting the state of a task

Each task can be set to a state:



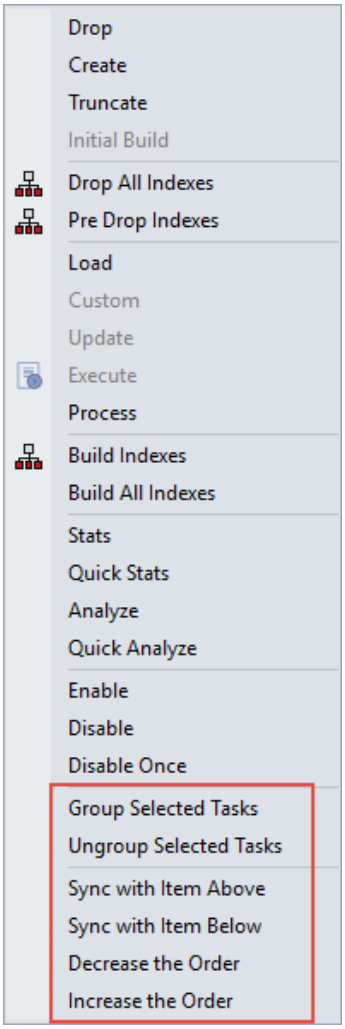
The following states are available:

State	Description
Enable	Job Task is enabled.
Disable	Job Task is disabled.
Disable Once	Job Task is disabled once and then reverts to enabled next time the Job is released by the Scheduler.

© Copyright, WhereScape Software Ltd All rights reserved.

Creating dependencies between tasks

You can create dependencies between tasks in the list by selecting one or more tasks and right-clicking to bring up the dependency options.



The following task dependency options are available from the menu:

Task Option	Description
Group Selected Tasks	Groups two or more selected tasks to have the same order value, allowing them to run in parallel if the maximum threads setting allows.
Ungroup Selected Tasks	Un-group selected tasks.
Sync with Item Above	Changes a selected task to have the same order value as the task above it, allowing them to run in parallel if the maximum threads setting allows.
Sync with Item Below	Changes a selected task to have the same order value as the task below it, allowing them to run in parallel if the maximum threads setting allows.
Decrease the Order	Changes a selected task to an order number one less than its current value. The task will now run immediately before it would have previously.
Increase the Order	Changes a selected task to an order number one more than its current value. The task will now run immediately after it would have previously.

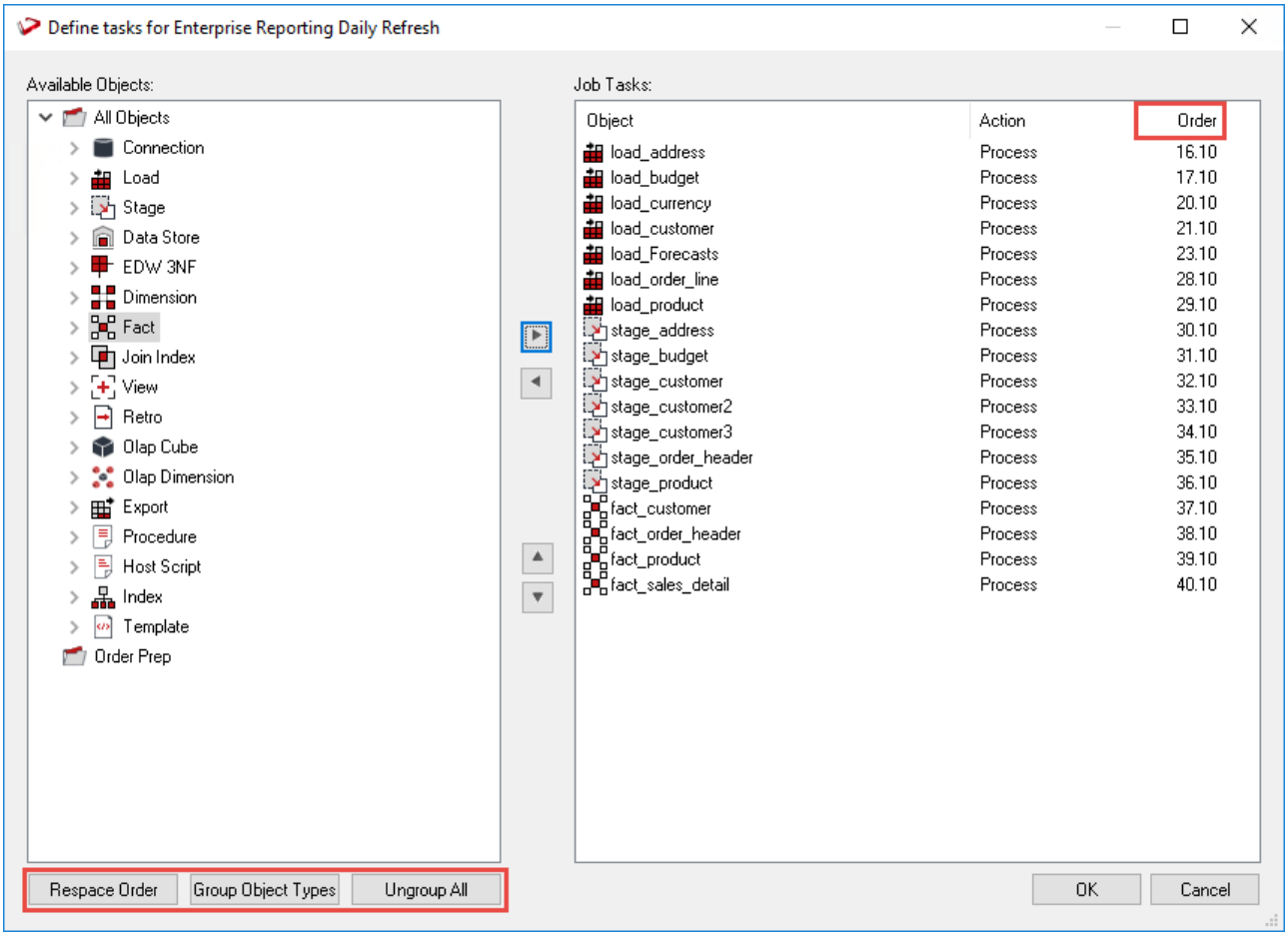
© Copyright, WhereScape Software Ltd All rights reserved.

Ordering or Grouping the tasks

The **Order** column shows the order in which the tasks are to be run, e.g. 20.20 If the two numbers are the same as another task then those tasks can run in parallel. If the two numbers are different then those tasks run sequentially. This is an initial definition of dependencies. These dependencies can be altered specifically once the job has been created.

Tasks can be moved up or down by selecting the task and clicking the **Move Up** or **Move Down** buttons.

To respace the order of the tasks; to group or ungroup object types, use the **buttons** at the bottom of the **Define tasks** window.



Button	Description
Respace Order	Click to respace the order numbers. The existing dependency structure and groupings are retained. The purpose of this button is simply to allow room between tasks to fit new tasks. For example, if we have two tasks that have an order of 20.19.5 and 20.20.6 and we want to add a task between these two tasks, we can click the **Respace Order** button to open up a gap between the two tasks.
Group Object Types	Click to put all objects of the same type into groups. For example, all load tables will be able to run in parallel, all dimensions, etc.
Ungroup All	Click to remove all groupings and make all tasks sequential. New groupings can be made by selecting a range of sequentially listed tasks in the left pane and using the right-click menu option **Group Selected Tasks**. Tasks that are grouped have the same first two numbers in the order and can execute at the same time if the job has multiple threads.

Upon completion of adding tasks, click **OK** to exit.

© Copyright, WhereScape Software Ltd All rights reserved.

Editing a Job

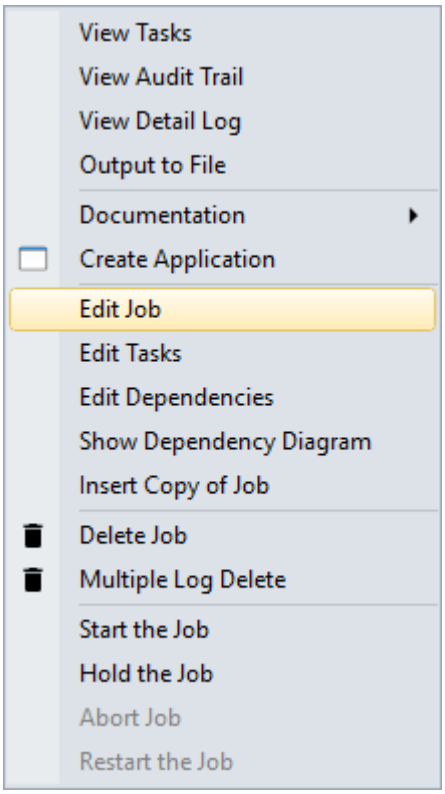
Once jobs have been created they can be edited.

Note

A job can only be edited when it is not in a running state and only if the job is a scheduled job. Completed jobs remain in the list but only logs remain.

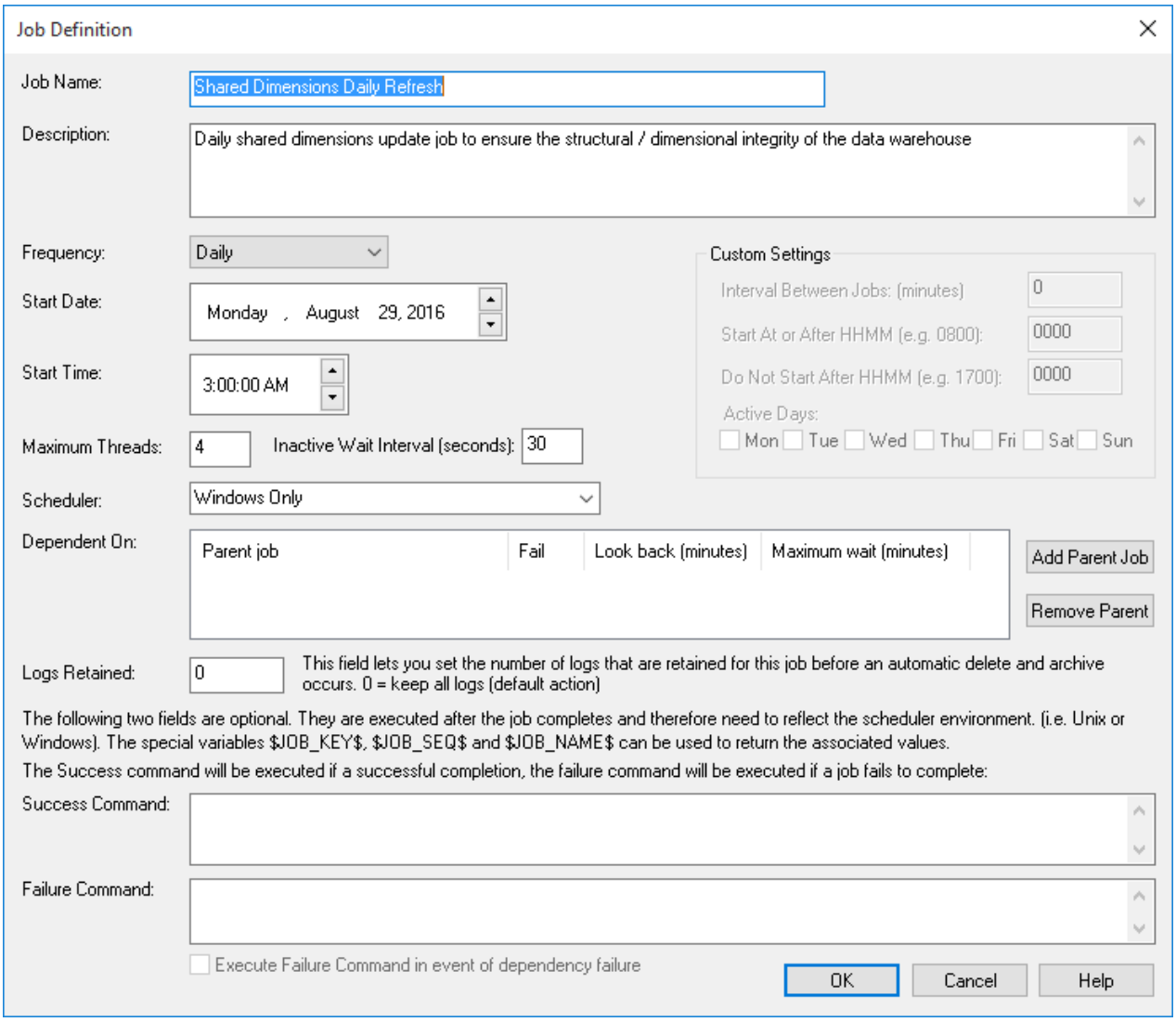
To edit a job

Select the job from the middle pane and then right-click to select **Edit Job** from the context menu.



The **Job Definition** is displayed.

© Copyright, WhereScape Software Ltd All rights reserved.



Edit the fields as required and click **OK**. The main fields are described in the following table:

Field	Description
Job Name	The Scheduler defaults to the next job number in the sequence. You can alter this to any alphanumeric. **Tip** Only alphanumerics, spaces and the underscore are supported in the name. **Warning** On some UNIX systems, long job names can cause jobs to be canceled (see Knowledge Base article 67), so where possible keep the name short.
Description	A description of the job
Frequency	When the job runs. The options available in the drop-down list box are: • Once Only - job is deleted on completion • Once and Hold - runs and puts another copy of the job on hold • Hold - puts the job on hold for manual release • Daily - runs the job daily • Custom - enables custom definition • Weekly - runs the job weekly • Monthly - runs the job monthly • Annually - runs the job annually

© Copyright, WhereScape Software Ltd All rights reserved.

Field	Description
Start Date and Start Time	The date and time for the job to start.
Max Threads	The maximum number of threads allocated to run the job, e.g, if some tasks can run in parallel then if more than one thread is allocated then they will run in parallel. **Caution** Users must be aware that adding additional threads to a job increases the communication overhead with the DB server containing the Data Warehouse repository. Each job thread holds a dedicated connection to the server for its lifetime.
Scheduler	Certain types of jobs will only run in a specific environment. For example, ODBC based loads can only be handled by the Windows' scheduler. It is possible to have multiple schedulers running. Select the desired scheduler from this drop-down. The valid options are:UNIX Preferred, UNIX Only, Windows Preferred, Windows Only, or the name of a specific scheduler can be entered (e.g. WIN0002)
Dependent On	A job can be dependent on the successful completion of one or more other jobs. Click the **Add Parent Job** button to select a job that this job will be dependent on. The maximum time to look back for parent job completion field prevents older iterations of the parent job as being identified as a completion. The maximum time to wait specifies how long to await a successful completion of the parent job. The action if that wait expires can also be set. Refer to the Job Dependency example in **Scheduling a Job** for details.
Logs Retained	Specify the number of logs to retain for the job. By default, all logs are retained. This field can be used to reduce the build up of scheduler logs by specifying a number of logs to retain.
Success command and Failure command	These are either UNIX or Windows shell commands depending on which scheduler is used. They are executed if the condition is met. Typically, these commands would mail or page on success or failure. **Notes** • The RED scheduler does not check return codes from called commands, scripts and programs. • It is recommended that all output from commands, scripts and programs is redirected to a log file. For example, add this to the end of any SUCCESS/FAILURE commands: >> c:\scheduler\success_failure_prod.log 2>&1

The following fields are available if a frequency of **Custom** is chosen:

Field	Description
Interval between jobs (Minutes)	Specify the number of minutes between iterations of the job. For example, to run a job every 30 minutes set this value to 30. If a job is to run only once but on selected days set this value to 1440 (daily)
Start at or after HHMM	The time that the job may run from. To run anytime set to 0000.
Do not start after HHMM	If multiple iterations are being done then this is the time after which a new iteration will not be started. For example, if a job is running every 10 minutes it will continue until this time is reached. To run till the end of day set to 2400.
Active on the days	Select each day of the week that the custom job is to be active on.

© Copyright, WhereScape Software Ltd All rights reserved.

Editing Tasks in a Job

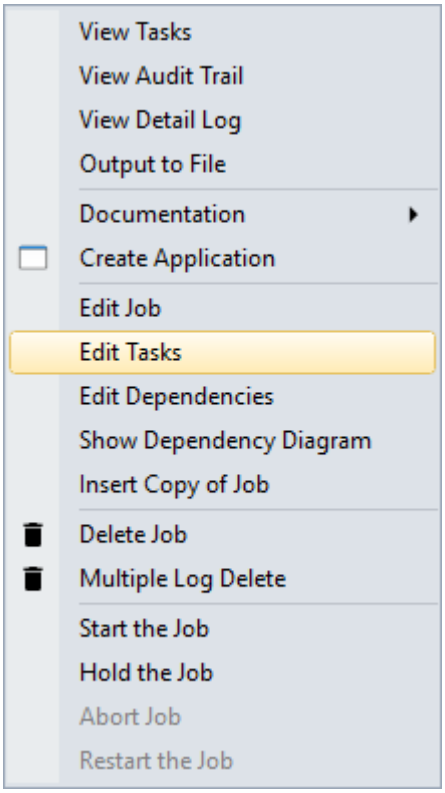
Once jobs have been created, you can edit their tasks.

Note

- A job can only be edited when it is not in a running state and only if the job is a scheduled job. Completed jobs remain in the list but only logs remain.
- **JOB TASK LIMIT -** There is maximum number of 999 tasks that can be added to a job.

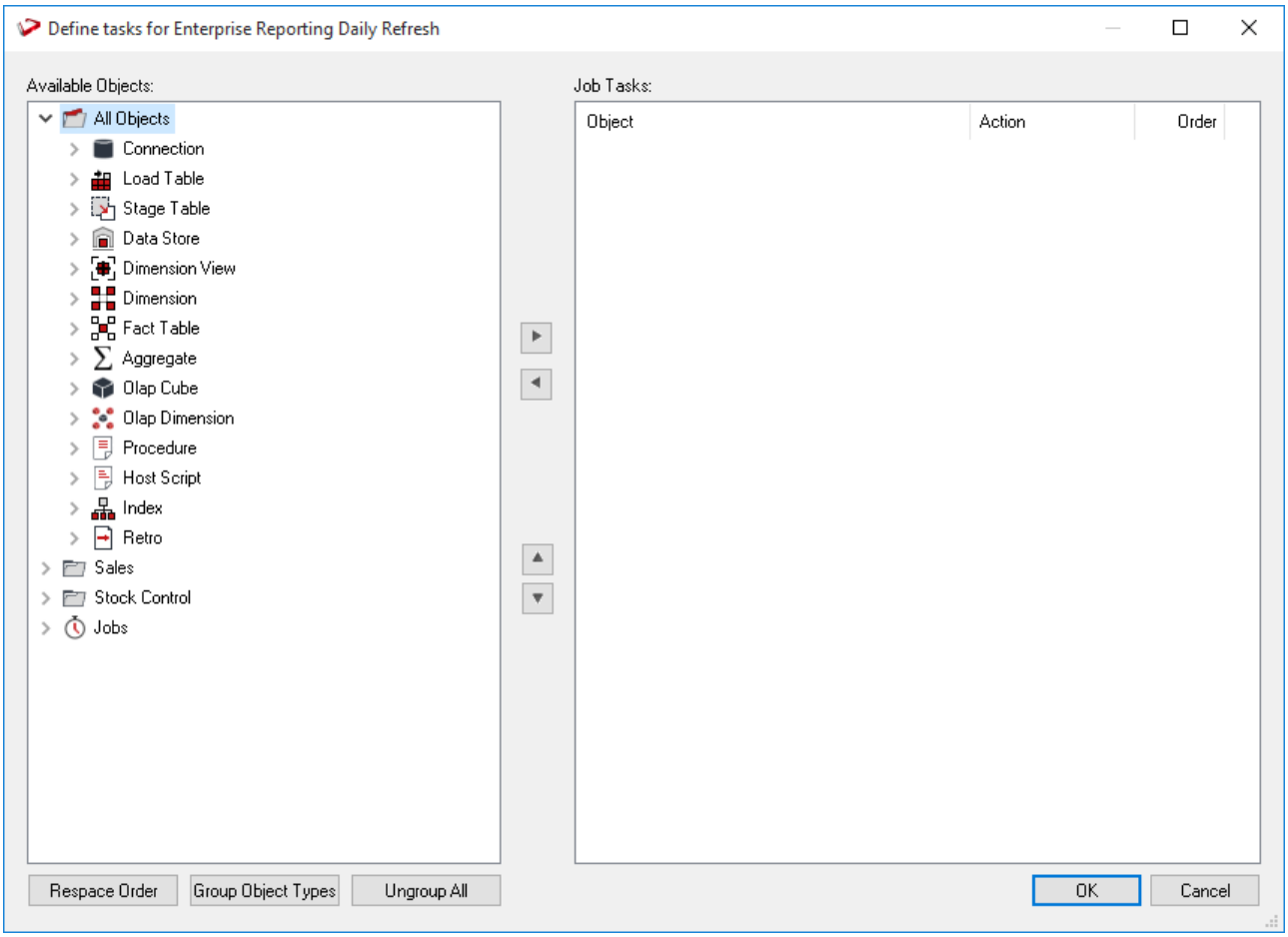
To edit the tasks of a job

Select the job from the middle pane and then right-click the job to select **Edit Tasks** from the context menu.



© Copyright, WhereScape Software Ltd All rights reserved.

The **Define tasks** window is displayed.



The screen has two main areas. The right pane shows the tasks to be run for this job and the left pane lists all the objects.

Adding a task

Double-click an object in the left pane to add it to the task list in the right pane. Normally objects, such as Load or Fact tables are scheduled rather than procedures.

Setting the action on a task

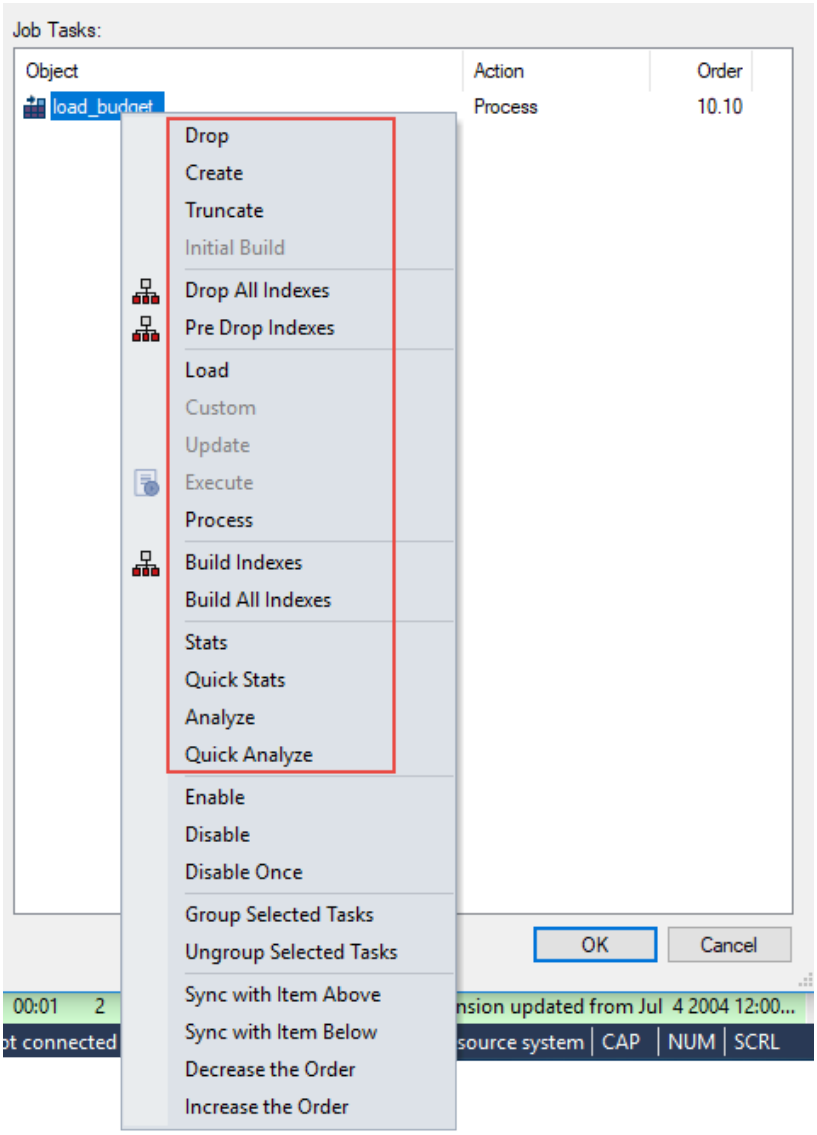
Each task can have a specific action that is to be performed on its object.

The default action for **load tables** is **process**. This means that when the task is actioned, it will drop any indexes that are due to be dropped, or have **pre-drop** set, then load the table and perform any post-load procedures or transformations and then re-create any dropped indexes.

The default action for all other tables is the same as above, except it will execute the **update** procedure rather than loading the table.

You can change the action on a task by right-clicking the task in the right pane. The menu options are shown below.

© Copyright, WhereScape Software Ltd All rights reserved.



The following task actions are available:

Action	Description
Drop	Drop table, view or index.
Create	Create table, view or index.
Truncate	Delete all rows from the table.
Initial Build	**Drop All Indexes** then **Custom** then **Build All Indexes**.
Drop All Indexes	Drop all indexes on the table.
Pre Drop Indexes	Drop all indexes on the table marked as "Pre Drop".
Load	Load the table (Load tables only).
Custom	Run the custom procedure on the table.
Update	Run the update procedure on the table.
Execute	Execute the procedure or host script.
Process	**Pre Drop Indexes** then **Update** and then **Build Indexes**.
Process and Statistics	**Process** then Default Stats as defined in the Table Properties > Statistics > Process and statistics method (DB2 only).
Build Indexes	Build the indexes on the table marked as "Pre Drop".
Build All Indexes	Build all indexes on the table.

© Copyright, WhereScape Software Ltd All rights reserved.

Action	Description
DB2: Analyze	Performs a custom analyze if defined in: • Table Properties > Statistics > Analyze else for tables performs: • **RUNSTATS ON TABLE tabname WITH DISTRIBUTION AND DETAILED INDEXES ALL** otherwise (for indexes): • **RUNSTATS ON TABLE indname FOR INDEXES**
DB2: Quick Analyze	Performs a custom analyze if defined in: • Table Properties > Statistics > Quick Analyze else for tables performs: • **RUNSTATS ON TABLE tabname** otherwise (for indexes): • **RUNSTATS ON TABLE indname FOR INDEXES**
DB2: Stats	Performs a custom stats if defined in: • Table Properties > Statistics > Stats else for tables performs: • **RUNSTATS ON TABLE tabname WITH DISTRIBUTION AND DETAILED INDEXES ALL** otherwise (for indexes): • **RUNSTATS ON TABLE indname FOR INDEXES**
DB2: Quick Stats	Performs a custom analyze if defined in: • Table Properties > Statistics > Quick Stats else for tables performs: • **RUNSTATS ON TABLE tabname** otherwise (for indexes): • **RUNSTATS ON TABLE indname FOR INDEXES**
Oracle: Analyze	Performs a custom analyze if defined in: • Tools > Options > Statistics > Table > Index Analyze Full otherwise performs: • **ANALYZE TABLE / INDEX name COMPUTE STATISTICS**
Oracle: Quick Analyze	Performs a custom analyze if defined in: • Tools > Options > Statistics > Table/ Index Analyze Quick otherwise performs: • **ANALYZE TABLE / INDEX name ESTIMATE STATISTICS SAMPLE 3 PERCENT**
Oracle: Stats	Performs a custom stats if defined in: • Tools > Options > Statistics > Table/Index Stats Full otherwise performs:

© Copyright, WhereScape Software Ltd All rights reserved.

Action	Description
	• **DBMS_STATS.GATHER_ TABLE / INDEX _STATS()** using the ownname and tabname / indname parameters with cascade.
Oracle: Quick Stats	Performs a custom analyze if defined in: • Tools > Options > Statistics > Table/Index Stats Quick otherwise performs: • **DBMS_STATS.GATHER_ TABLE / INDEX _STATS()** using the ownname and tabname/indname parameters with cascade and estimate_percent = 3.
SQL Server: Analyze	Performs: • **UPDATE STATISTICS name WITH FULLSCAN**
SQL Server: Quick Analyze	Performs: • **UPDATE STATISTICS name WITH SAMPLE 3 PERCENT**
SQL Server: Stats	Same as Analyze.
SQL Server: Quick Stats	Same as Quick Analyze.

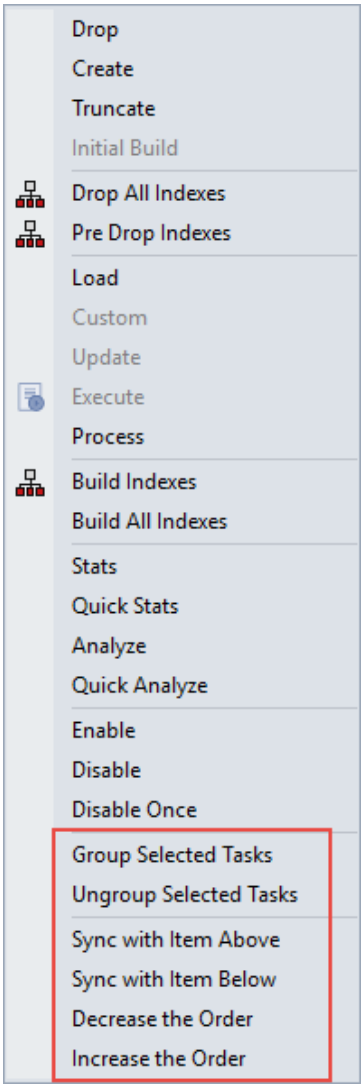
Note

Not all actions are available on all object types.

© Copyright, WhereScape Software Ltd All rights reserved.

Creating dependencies between tasks

You can create dependencies between tasks in the list by selecting one or more tasks and right-clicking to bring up the dependency options.



The following task dependency options are available from the menu:

Task Option	Description
Group Selected Tasks	Groups two or more selected tasks to have the same order value, allowing them to run in parallel if the maximum threads setting allows.
Ungroup Selected Tasks	Un-group selected tasks.
Sync with Item Above	Changes a selected task to have the same order value as the task above it, allowing them to run in parallel if the maximum threads setting allows.
Sync with Item Below	Changes a selected task to have the same order value as the task below it, allowing them to run in parallel if the maximum threads setting allows.
Decrease the Order	Changes a selected task to an order number one less than its current value. The task will now run immediately before it would have previously.
Increase the Order	Changes a selected task to an order number one more than its current value. The task will now run immediately after it would have previously.

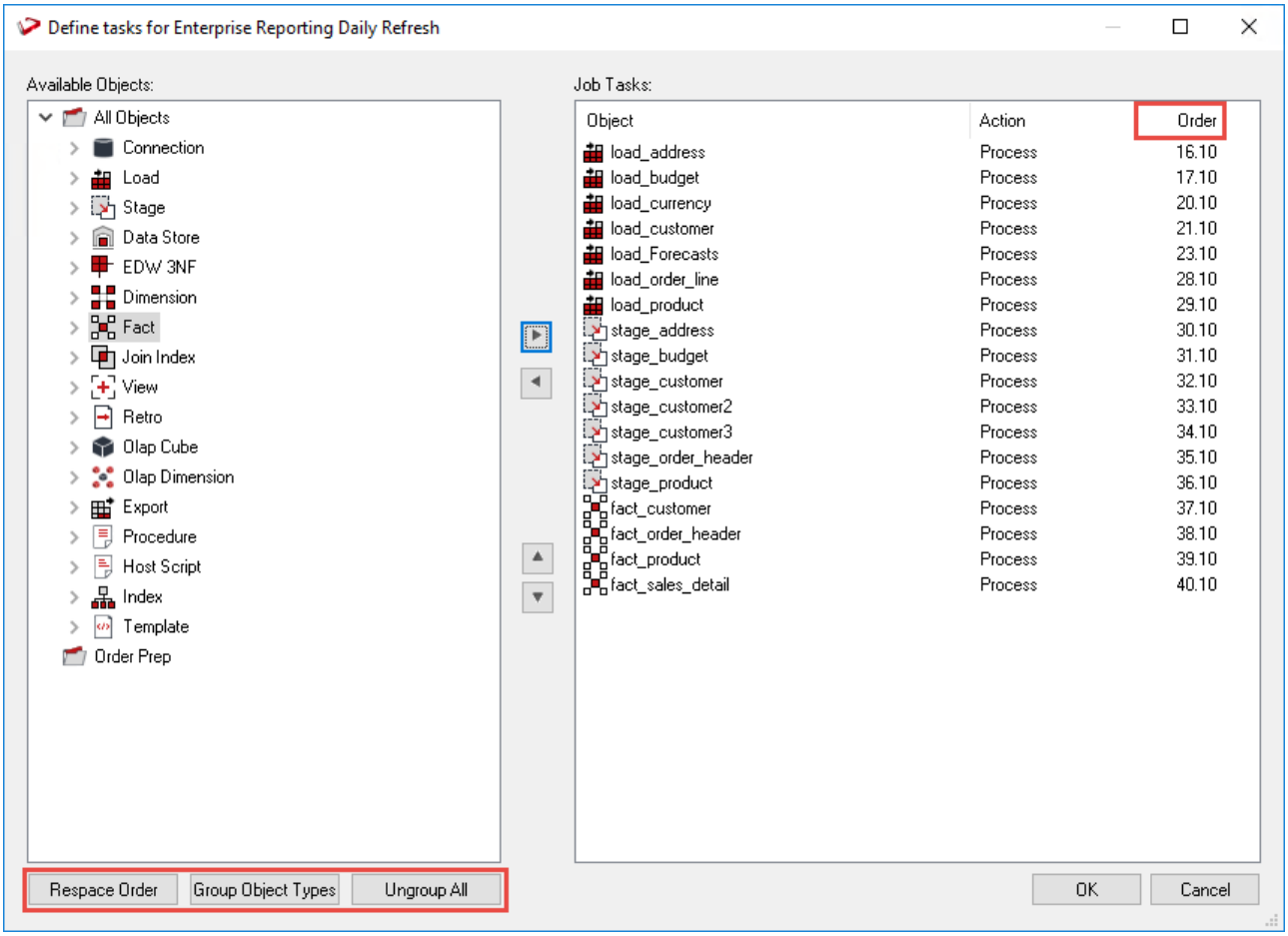
© Copyright, WhereScape Software Ltd All rights reserved.

Ordering or Grouping the tasks

The **Order** column shows the order in which the tasks are to be run, e.g. 20.20 If the two numbers are the same as another task then those tasks can run in parallel. If the two numbers are different then those tasks run sequentially. This is an initial definition of dependencies. These dependencies can be altered specifically once the job has been created.

Tasks can be moved up or down by selecting the task and clicking the **Move Up** or **Move Down** buttons.

To respace the order of the tasks; to group or ungroup object types, use the **buttons** at the bottom of the **Define tasks** window.



Button	Description
Respace Order	Click to respace the order numbers. The existing dependency structure and groupings are retained. The purpose of this button is simply to allow room between tasks to fit new tasks. For example, if we have two tasks that have an order of 20.19.5 and 20.20.6 and we want to add a task between these two tasks, we can click the **Respace Order** button to open up a gap between the two tasks.
Group Object Types	Click to put all objects of the same type into groups. For example, all load tables will be able to run in parallel, all dimensions, etc.
Ungroup All	Click to remove all groupings and make all tasks sequential. New groupings can be made by selecting a range of sequentially listed tasks in the left pane and using the right-click menu option **Group Selected Tasks**. Tasks that are grouped have the same first two numbers in the order and can execute at the same time if the job has multiple threads.

Upon completion of adding tasks, click **OK** to exit.

© Copyright, WhereScape Software Ltd All rights reserved.

Editing Task Dependencies

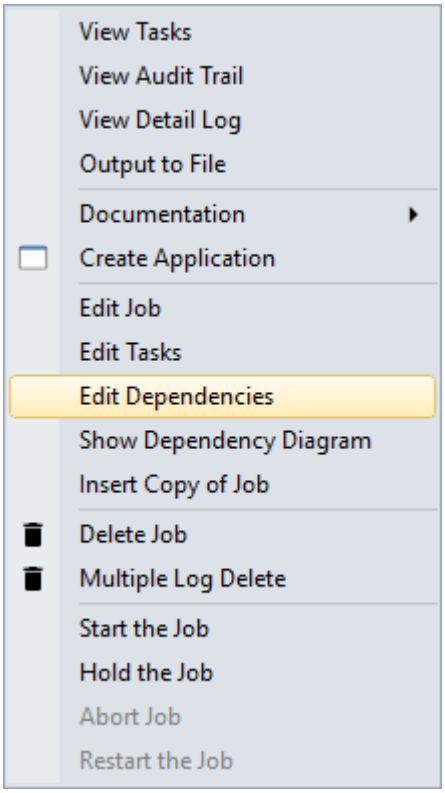
Once jobs have been created they can be edited.

Note

A job can only be edited when it is not in a running state and only if the job is a scheduled job. Completed jobs remain in the list but only logs remain.

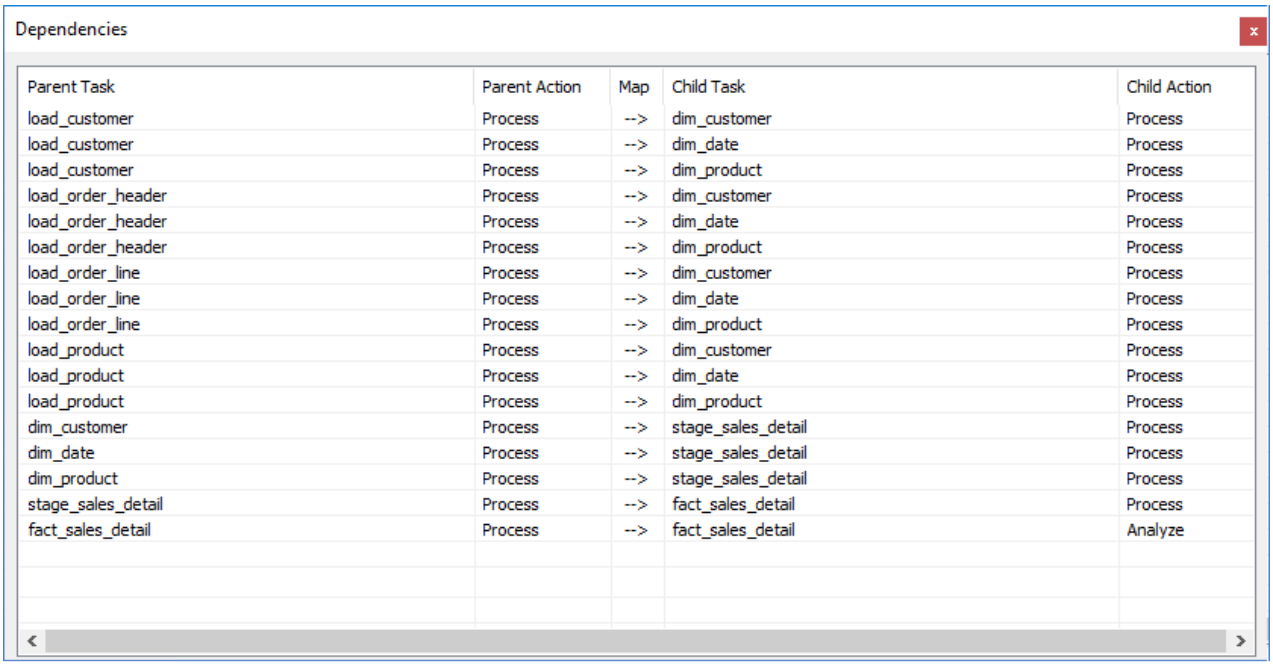
To edit task dependencies

Select the job from the middle pane and then right-click to select **Edit Dependencies** from the context menu.



The **Dependencies** window is displayed, showing the dependencies between the tasks of the job. The list consists of **Parent Tasks** on the left and **Child Tasks** on the right. A child task is thus dependent on its parent task in that it cannot run until its parent has run.

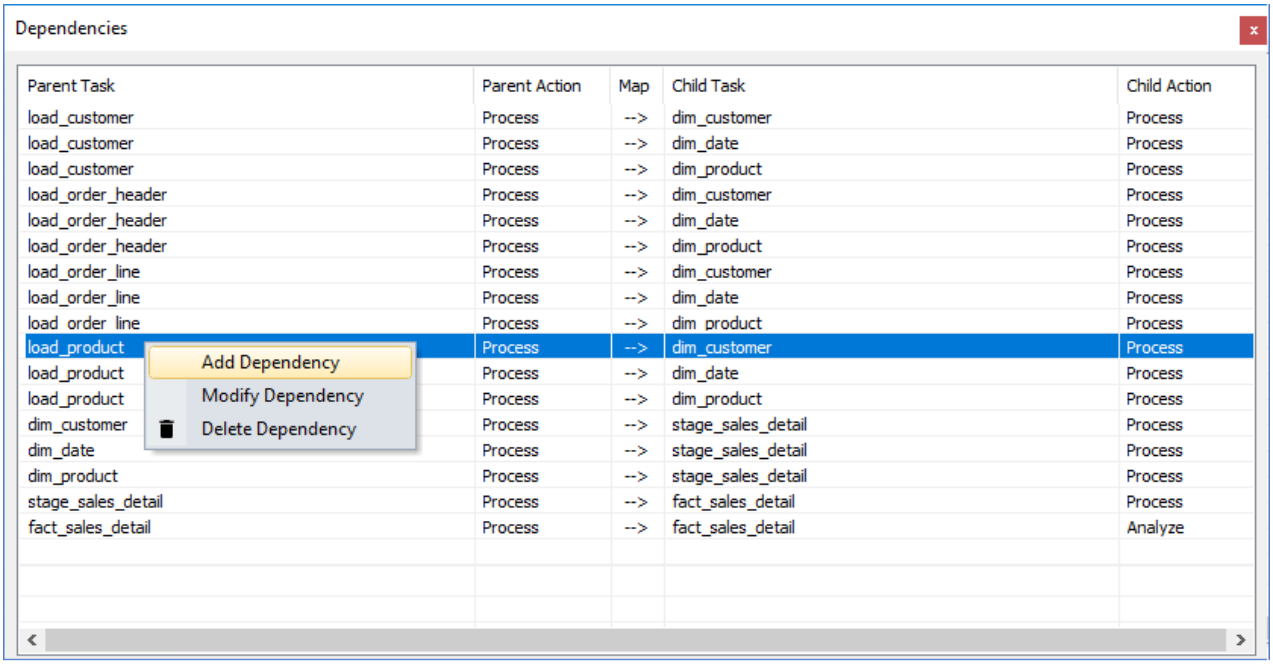
© Copyright, WhereScape Software Ltd All rights reserved.



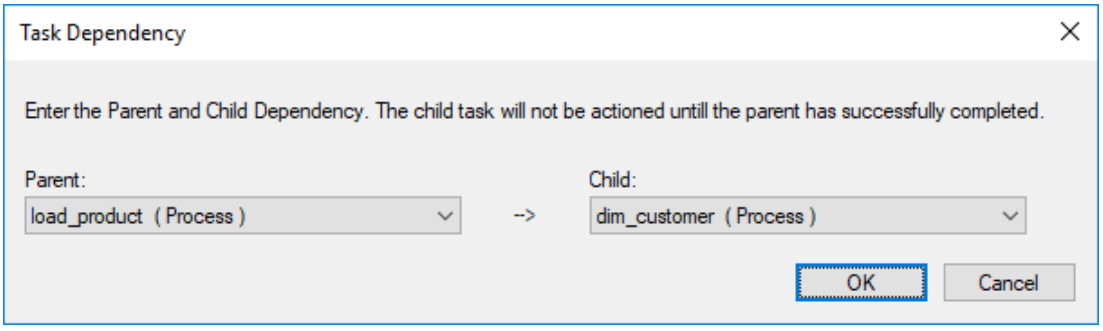
Edit the dependencies and close the window.

To add a task dependency

To add a task dependency, right-click anywhere in the **Dependencies** window and select **Add Dependency**.



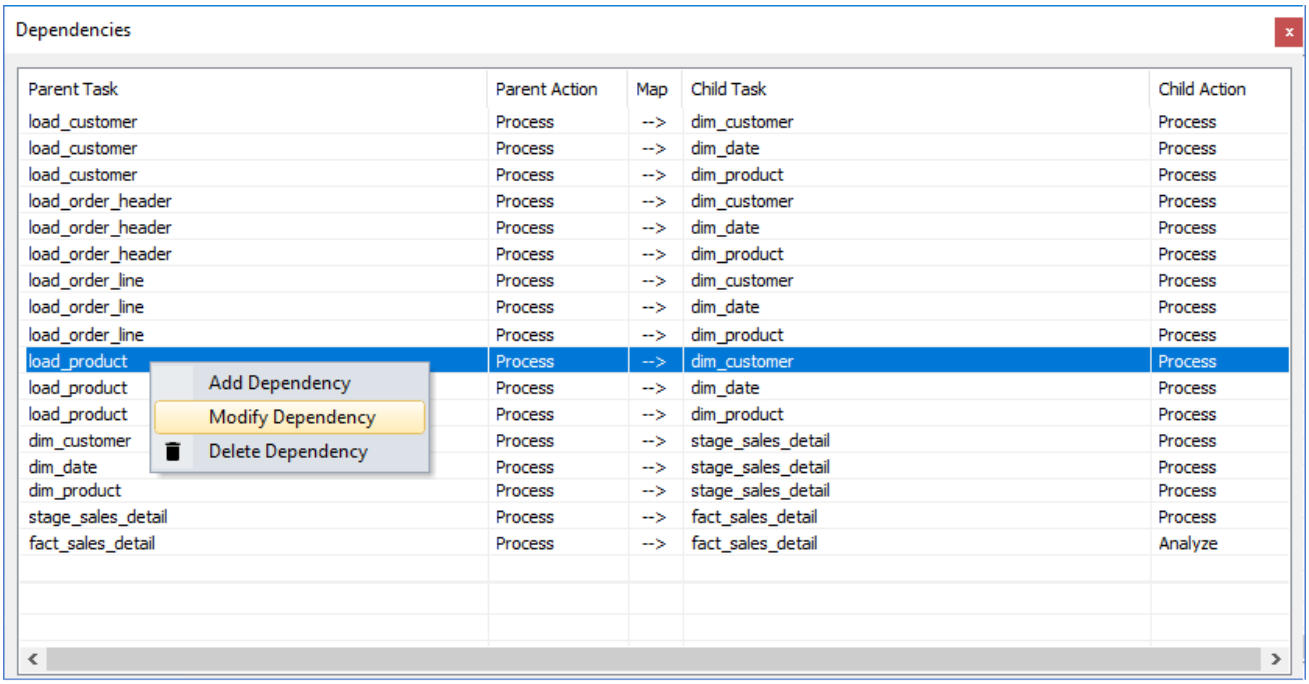
Select the **Parent** and **Child** tasks from the drop-down lists to create the dependency and then click **OK**.



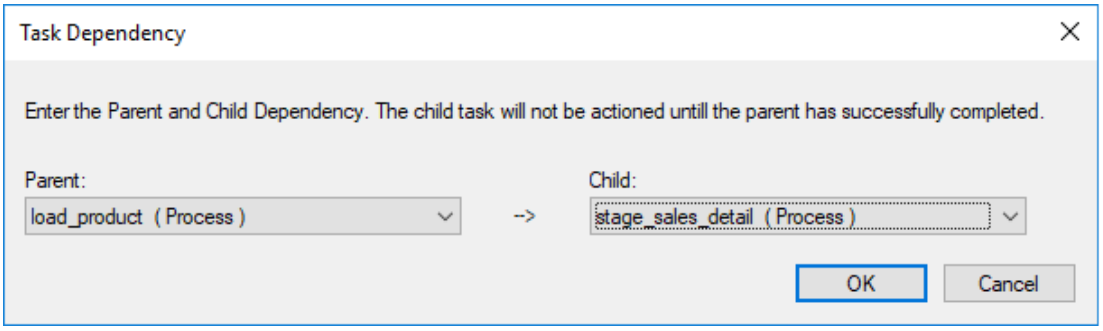
© Copyright, WhereScape Software Ltd All rights reserved.

To modify a task dependency

To modify a task dependency, right-click the dependency in the **Dependencies** window and select **Modify Dependency**.



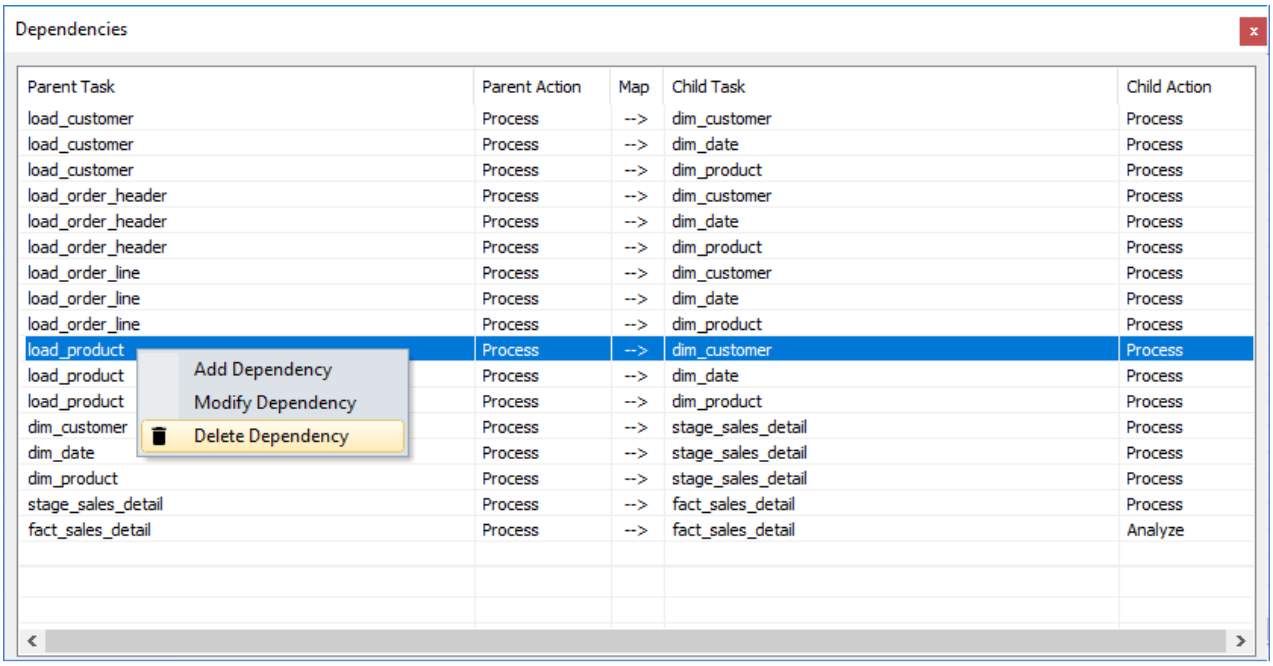
Change the **Parent** and **Child** tasks to modify the dependency and then click **OK**.



© Copyright, WhereScape Software Ltd All rights reserved.

To delete a task dependency

To delete a task dependency, right-click the dependency in the **Dependencies** window and select **Delete Dependency**.

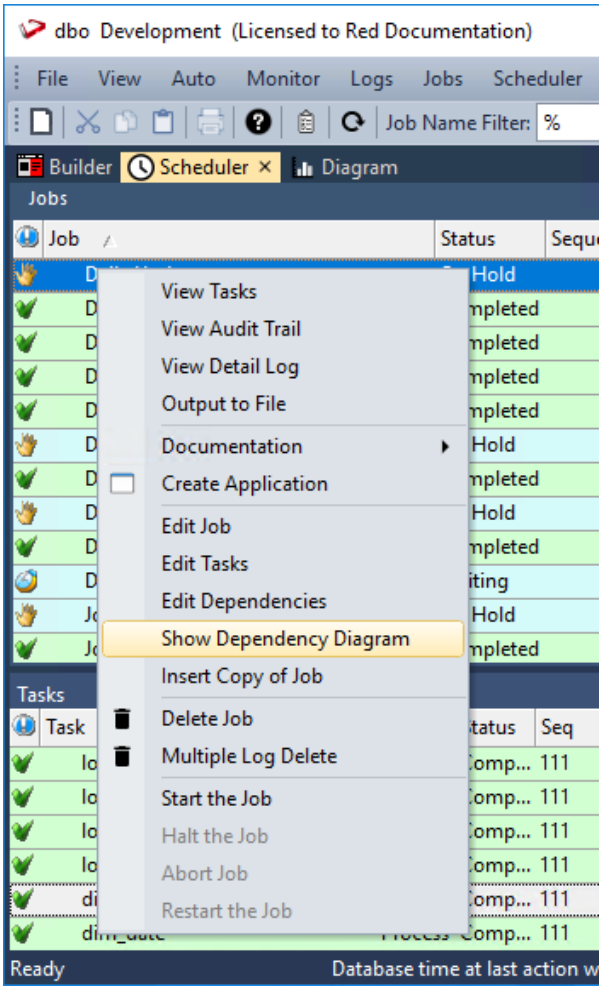


The dependency is deleted without warning.

© Copyright, WhereScape Software Ltd All rights reserved.

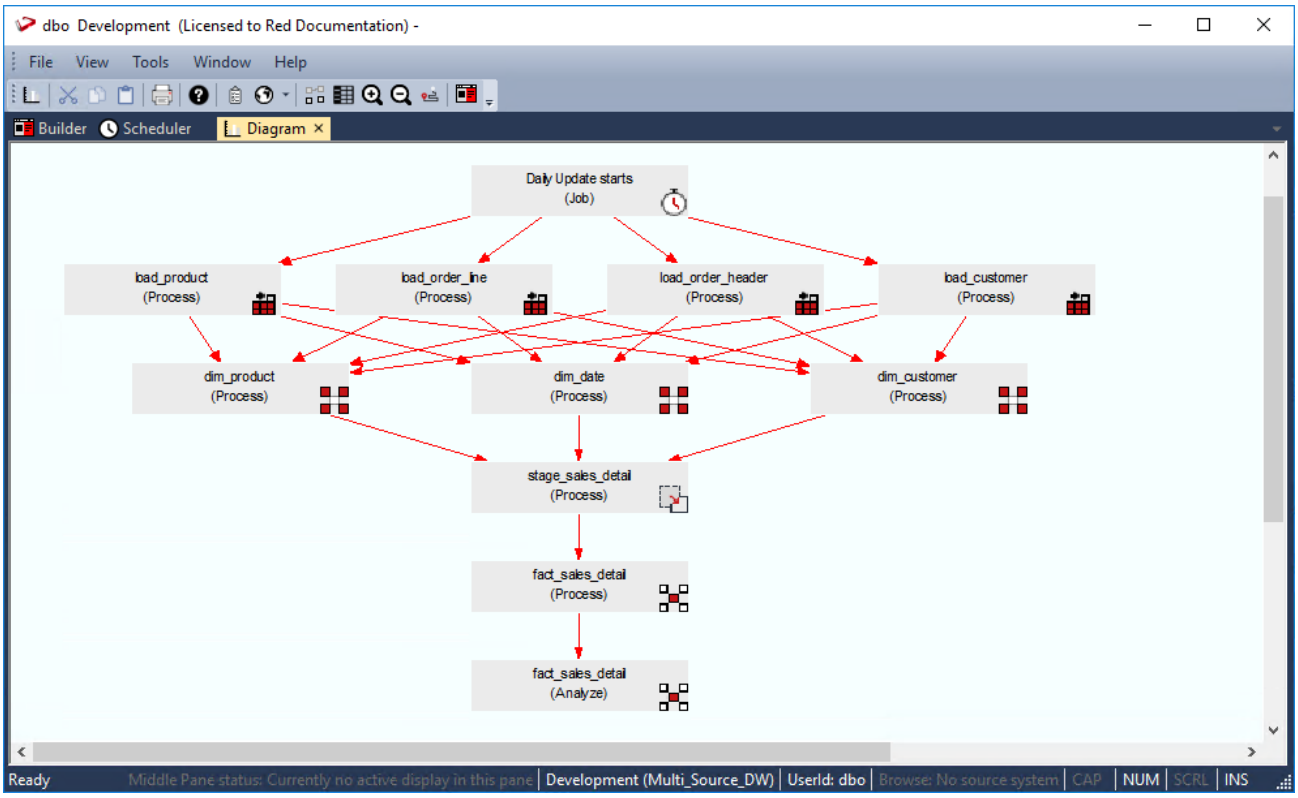
Show Dependencies Diagram

Select the **Show Dependency Diagram** option from the right-click menu of any job, to see all job dependencies displayed as a Diagram from RED's **Diagram** view tab.



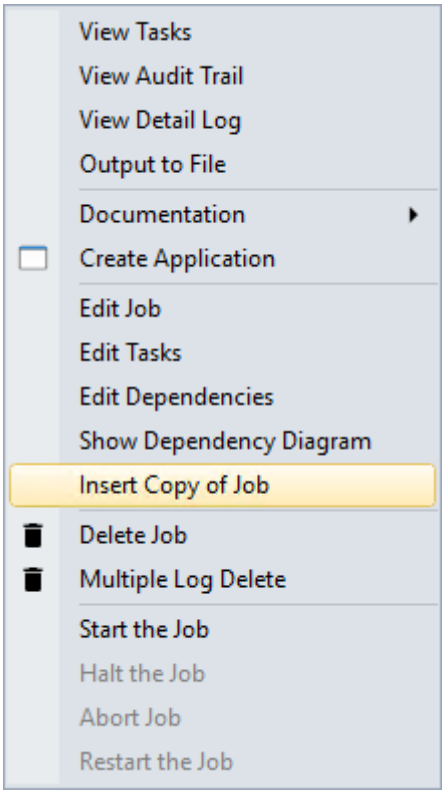
© Copyright, WhereScape Software Ltd All rights reserved.

Job Dependency Diagram view



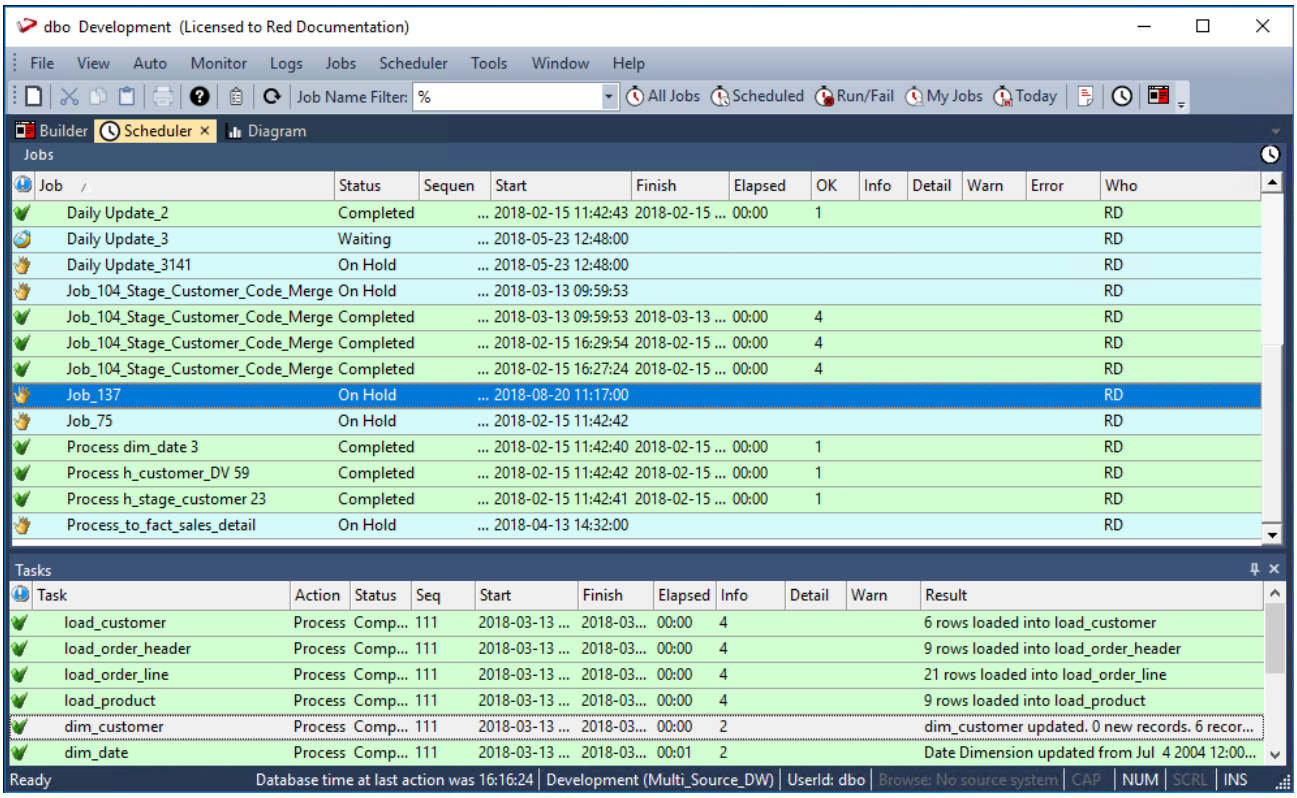
Inserting a Copy of a Job

A copy of a job can be inserted by right-clicking the job and choosing **Insert Copy of Job** from the context menu.



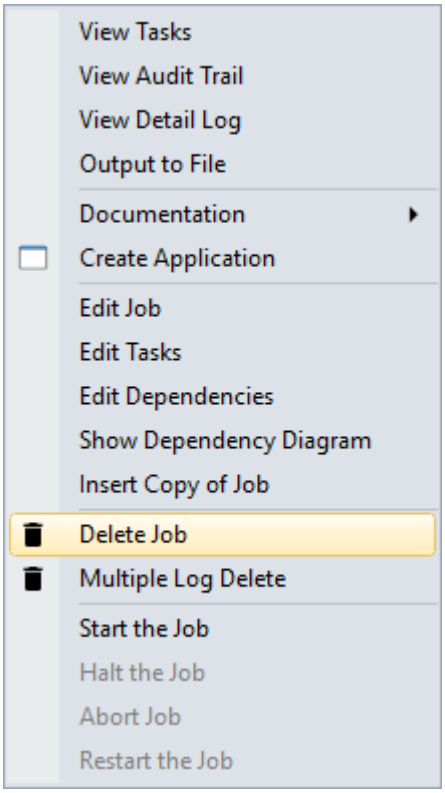
© Copyright, WhereScape Software Ltd All rights reserved.

The new job is immediately visible and the Status is **On Hold**.



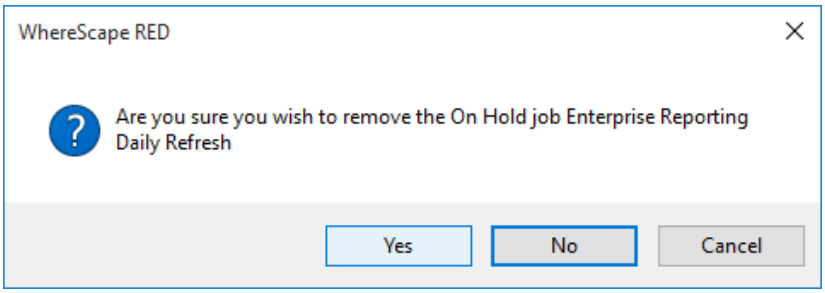
Deleting a Job

A job can be deleted by right-clicking the job in the Scheduler window and then choosing **Delete Job** from the context menu.



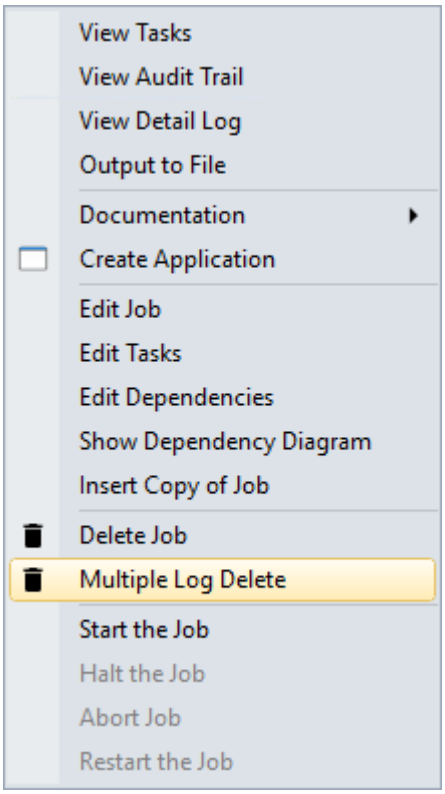
© Copyright, WhereScape Software Ltd All rights reserved.

A confirmation prompt is displayed; click **Yes** to delete.



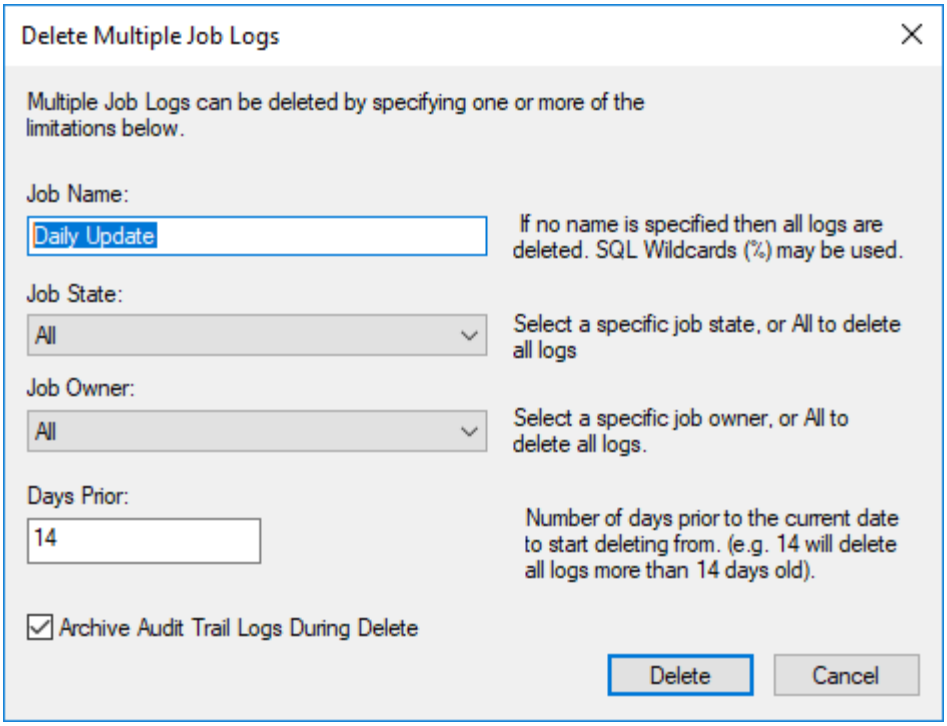
Deleting Job Logs

Multiple job logs can be deleted by right-clicking a job in the Scheduler window and then choosing **Multiple Log Delete** from the context menu.

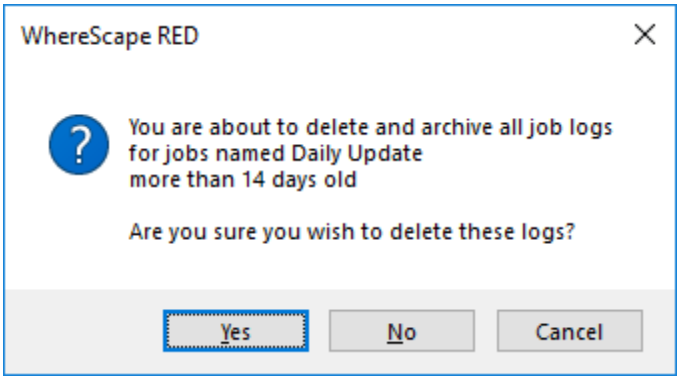


© Copyright, WhereScape Software Ltd All rights reserved.

The **Delete Multiple Job Logs** window is displayed. Select or enter the appropriate options to delete the range of job logs required.



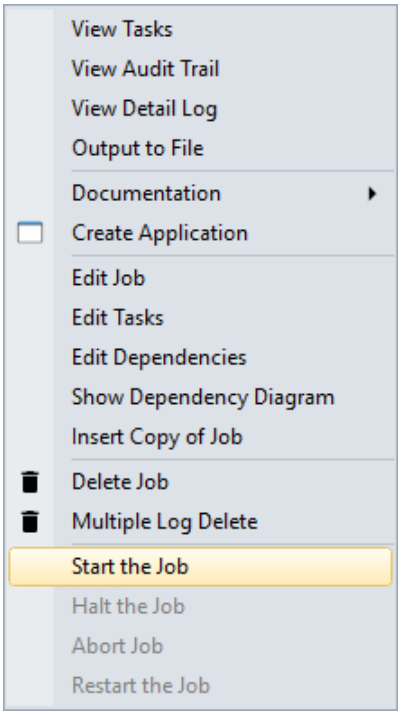
A confirmation prompt is displayed. Click **Yes** to delete.



© Copyright, WhereScape Software Ltd All rights reserved.

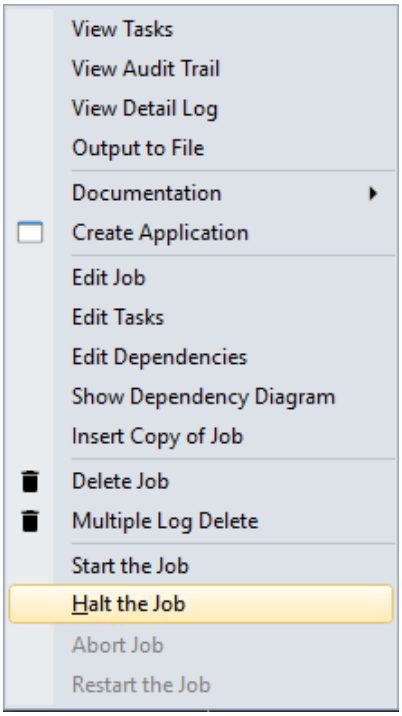
Starting a Job

A job can be started by right-clicking the job in the Scheduler window and then choosing **Start the Job** from the context menu.



Halting a Job

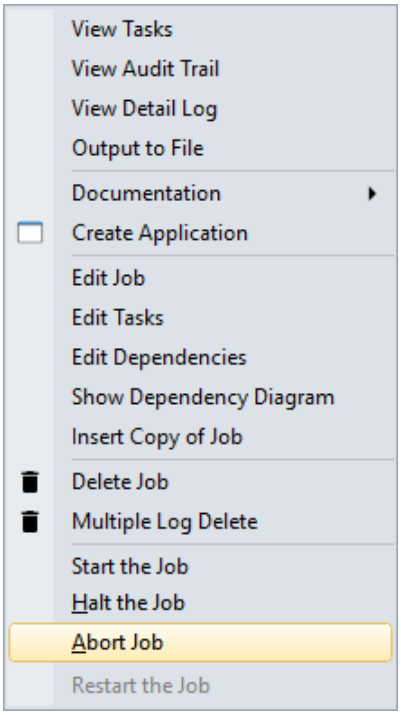
A job can be halted by right-clicking the job in the Scheduler window and then choosing **Halt the Job** from the context menu.



© Copyright, WhereScape Software Ltd All rights reserved.

Aborting a Job

A job can be aborted by right-clicking the job in the Scheduler window and choosing **Abort Job** from the context menu.



Once in this state, a job cannot be restarted. The job now only exists as a log of what occurred and is no longer regarded as a job.

Effects of Aborting a Job

The effects of aborting a job depend on the Operating System the Scheduler is running on, the target platform and the action being run.

Note

Rolling back the current transaction may take a considerable amount of time.

Windows Scheduler

SQL Server

Object	Action	Effect
Load	Native Load, SSIS Load	Does not stop process
Load	ODBC Load, Database Link Load, File/Script Windows Load	Stops process
All objects using Procedures	Update	Stops process and rolls back current transaction
Cube	Update	Does not stop process

Oracle

Object	Action	Effect
Load	Database Link Load	Stops process
Load	All other loads	Does not stop process
All objects using Procedures	Update	Stops process and rolls back current transaction

© Copyright, WhereScape Software Ltd All rights reserved.

Object	Action	Effect
Cube	Update	Does not stop process

DB2

Object	Action	Effect
Load	All Loads	Does not stop process
All objects using Procedures	Update	Stops process and rolls back current transaction
Cube	Update	Does not stop process

Greenplum (target)

Load and update processes are not stopped for all objects.

Hive (target)

Load and update processes are not stopped for all objects.

Netezza (target)

Load and update processes are not stopped for all objects.

PDW (target)

Object	Action	Effect
Load	Native Load, SSIS Load	Does not stop process
Load	ODBC Load, Database Link Load, File/Script Windows Load	Stops process
All objects using Procedures (Local targets)	Update	Stops process and rolls back current transaction
All objects using Procedures (remote targets)	Update	Does not stop process
Cube	Update	Does not stop process

Unix Scheduler

Oracle

Object	Action	Effect
Load	Database Link Load	Stops process and rolls back current transaction
Load	File/Script Unix Load	Does not stop process

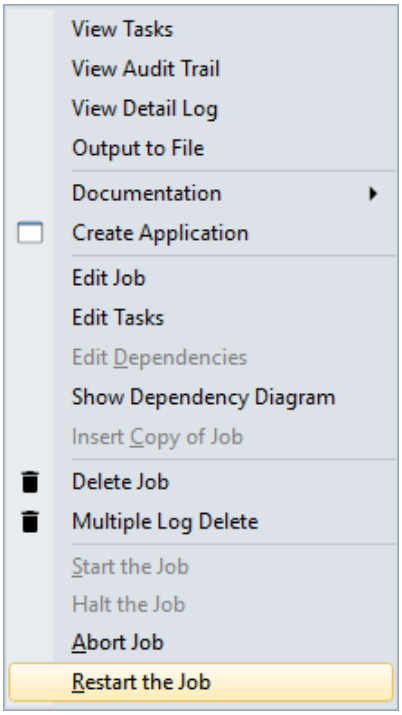
Hive (target)

File/Hadoop Load processes are not stopped on Hive target platforms.

© Copyright, WhereScape Software Ltd All rights reserved.

Restarting a Job

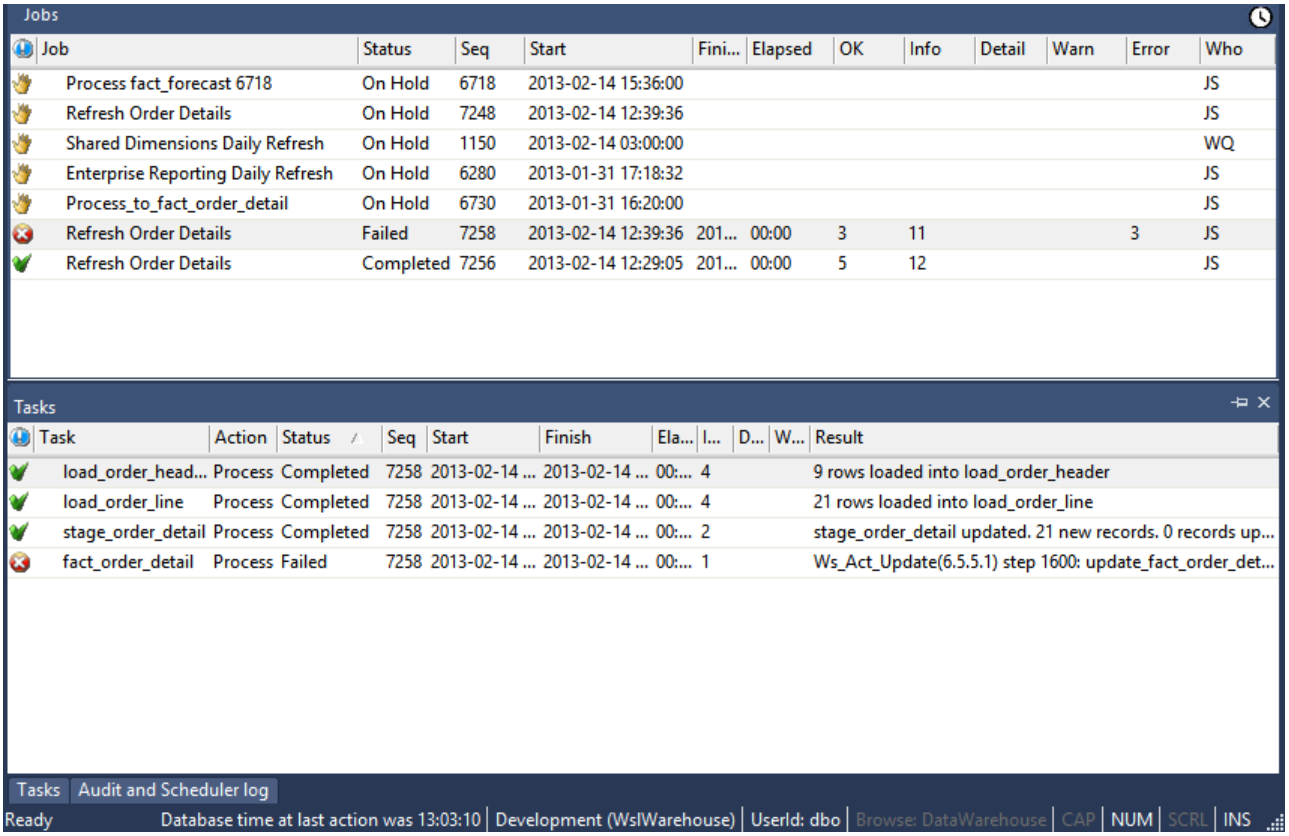
A job can be restarted by right-clicking the job in the Scheduler window and choosing **Restart the Job** from the context menu.



Before restarting a job, it is possible to edit the **status** of the job tasks so that only certain tasks will be **run again** or be **skipped over**.

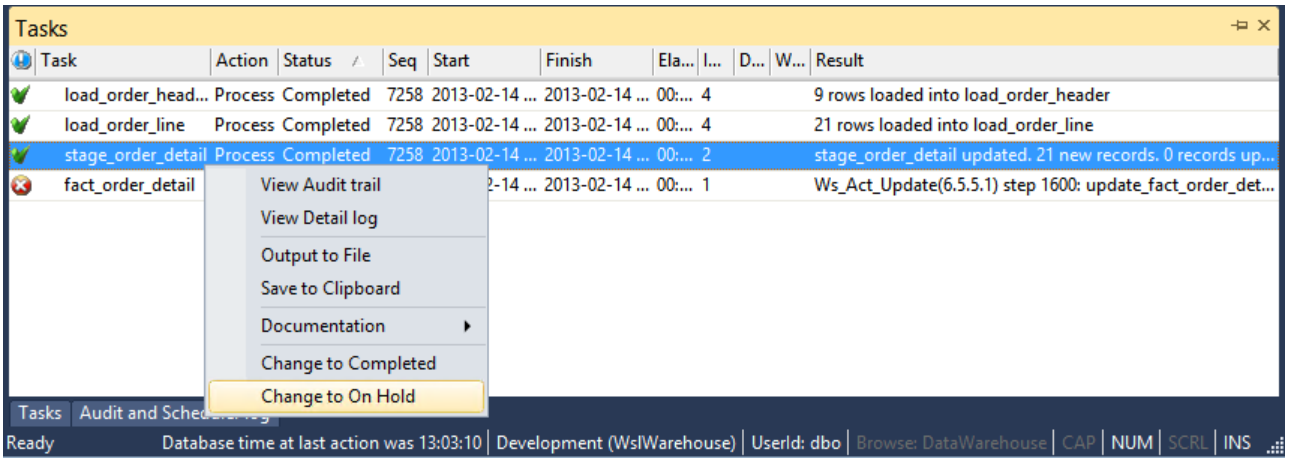
To run a task again

View the job tasks by double-clicking on the failed job. The tasks will be displayed in the bottom pane.

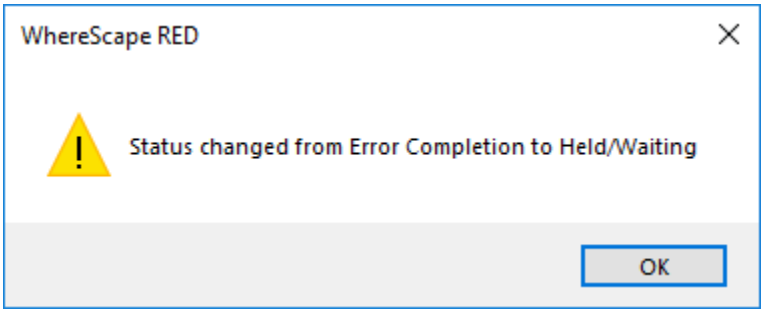


© Copyright, WhereScape Software Ltd All rights reserved.

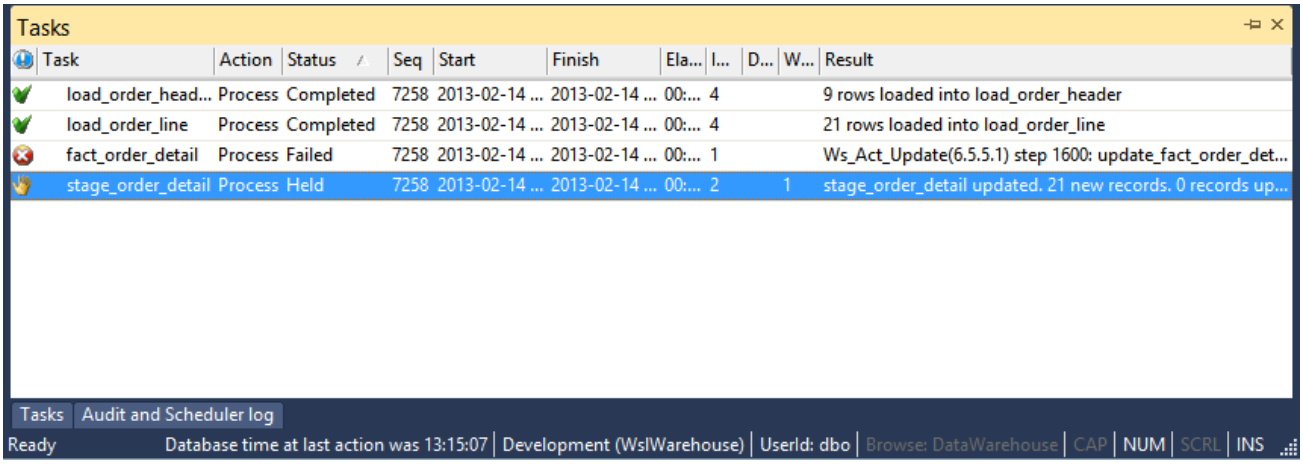
To rerun a task, right-click the completed task and select **Change to On Hold**



Click **OK** on the message dialog.



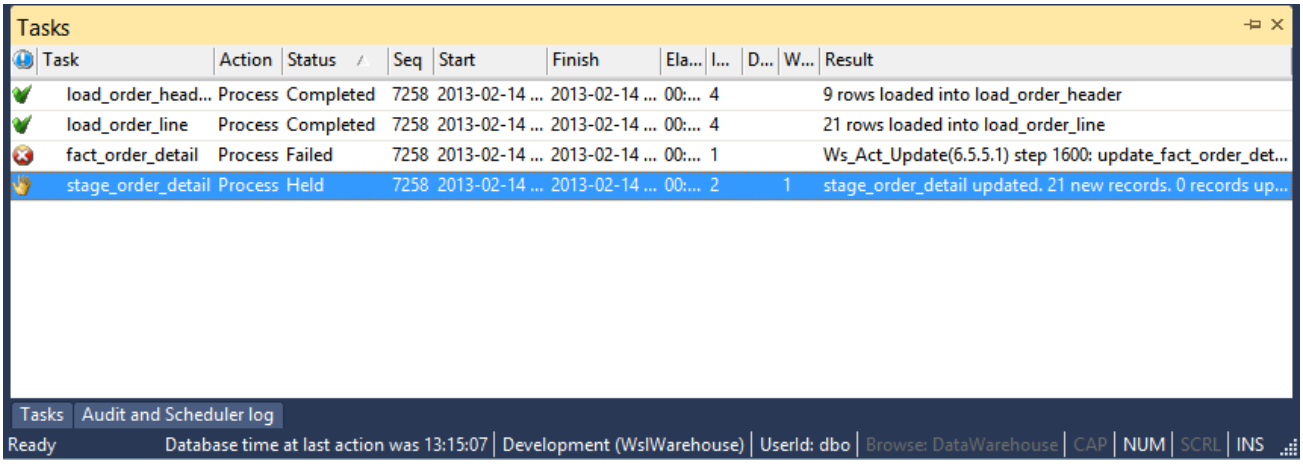
Double-click on the job again to display the tasks. You will see that the selected task now has a status of **Held** and will thus be rerun when you restart the job.



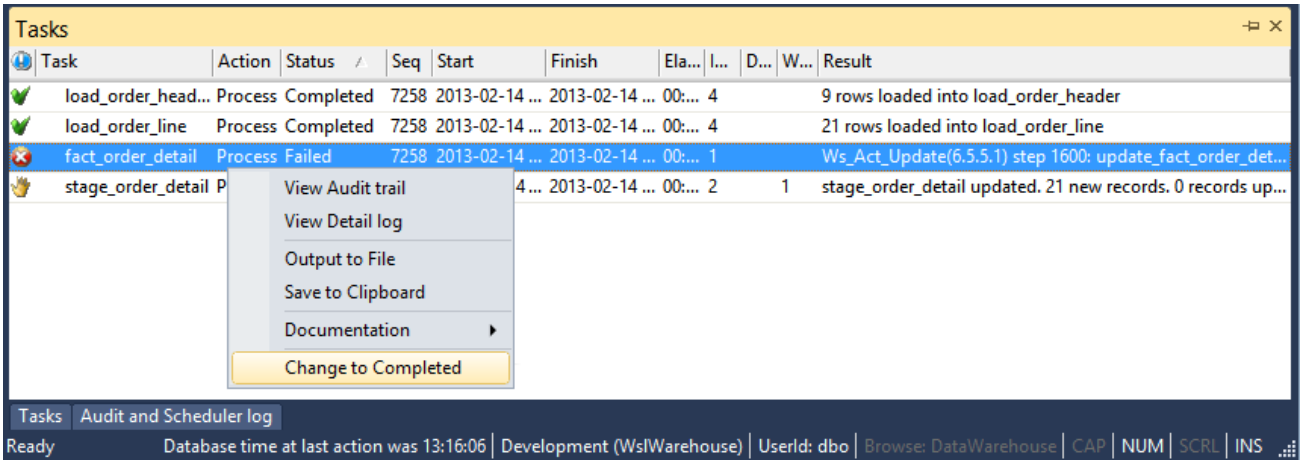
© Copyright, WhereScape Software Ltd All rights reserved.

To skip over a task

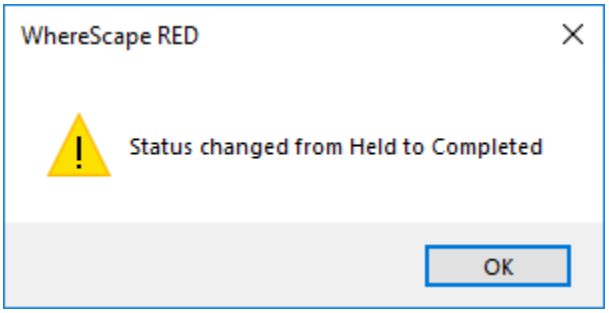
View the job tasks by double-clicking on the failed job. The tasks will be displayed in the bottom pane.



To skip over a task, right-click the task and select **Change to Completed**.

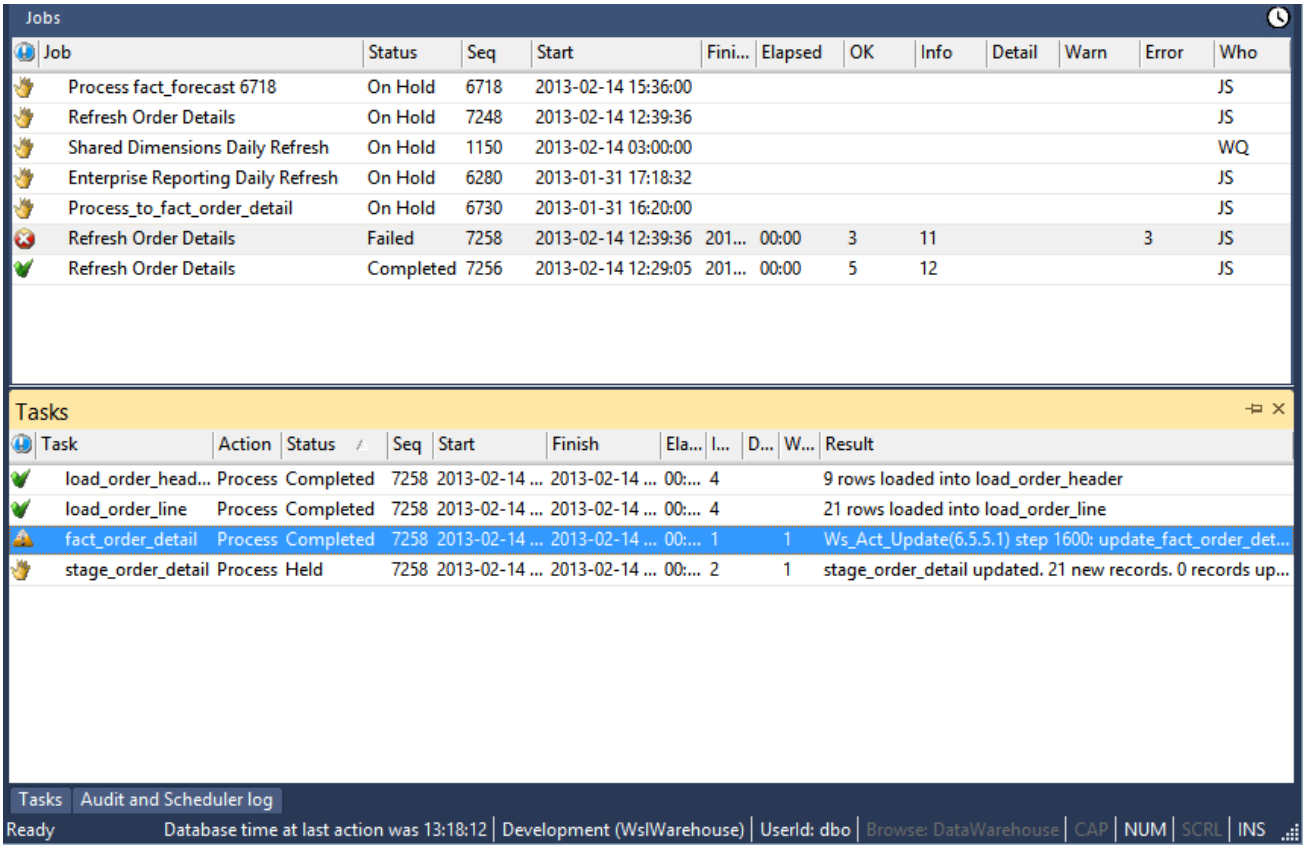


Click **OK** on the message dialog.



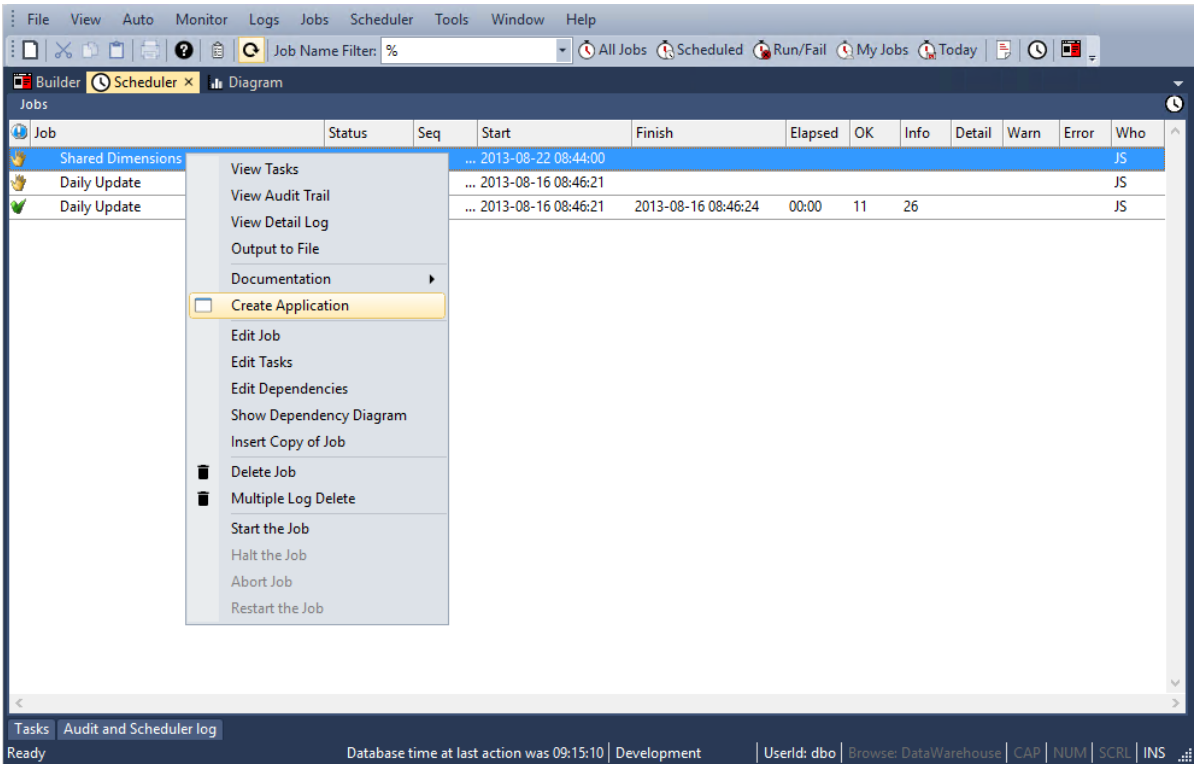
© Copyright, WhereScape Software Ltd All rights reserved.

Double-click the job again to display the tasks. The selected task now has a status of **Completed** and will thus be skipped when you restart the job.



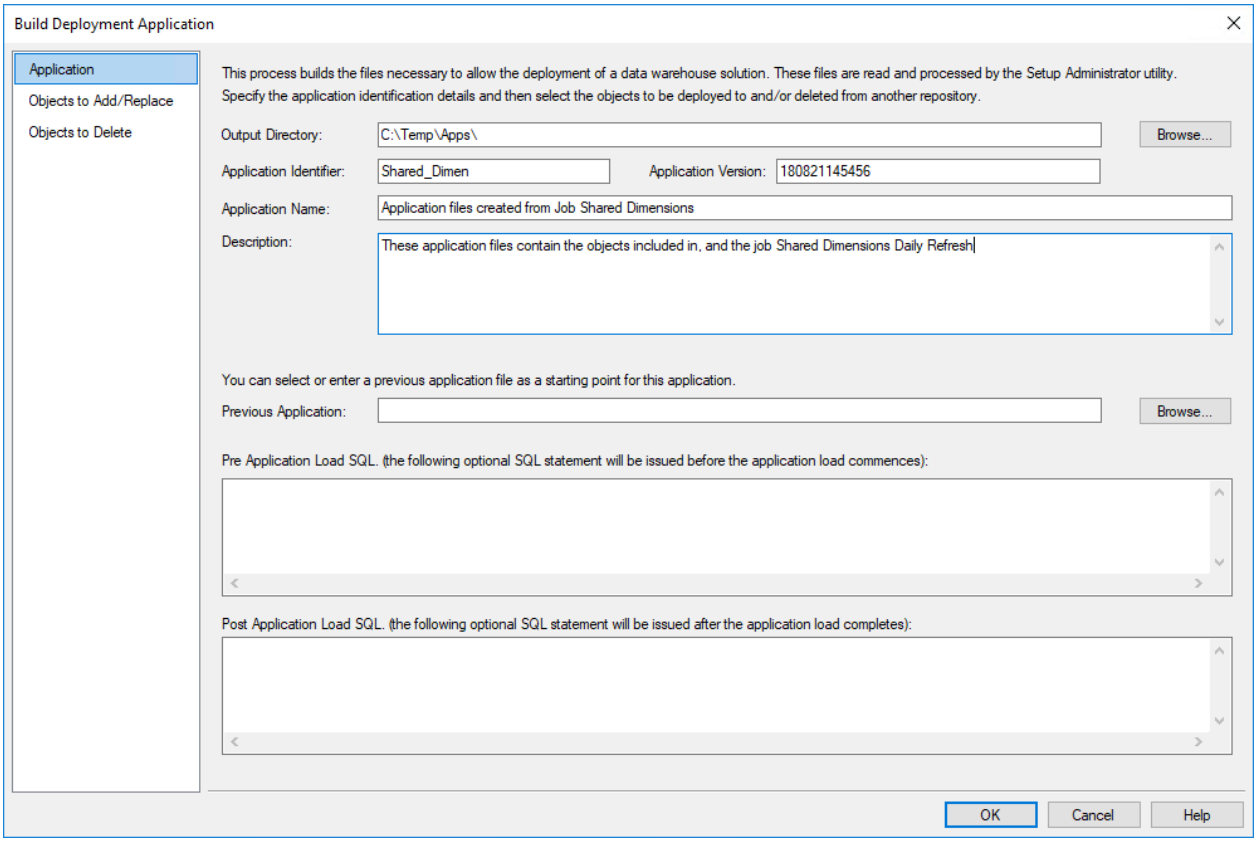
Creating an Application from a Job

1. Right-click the job in the Scheduler window and then select **Create Application** from the context menu.

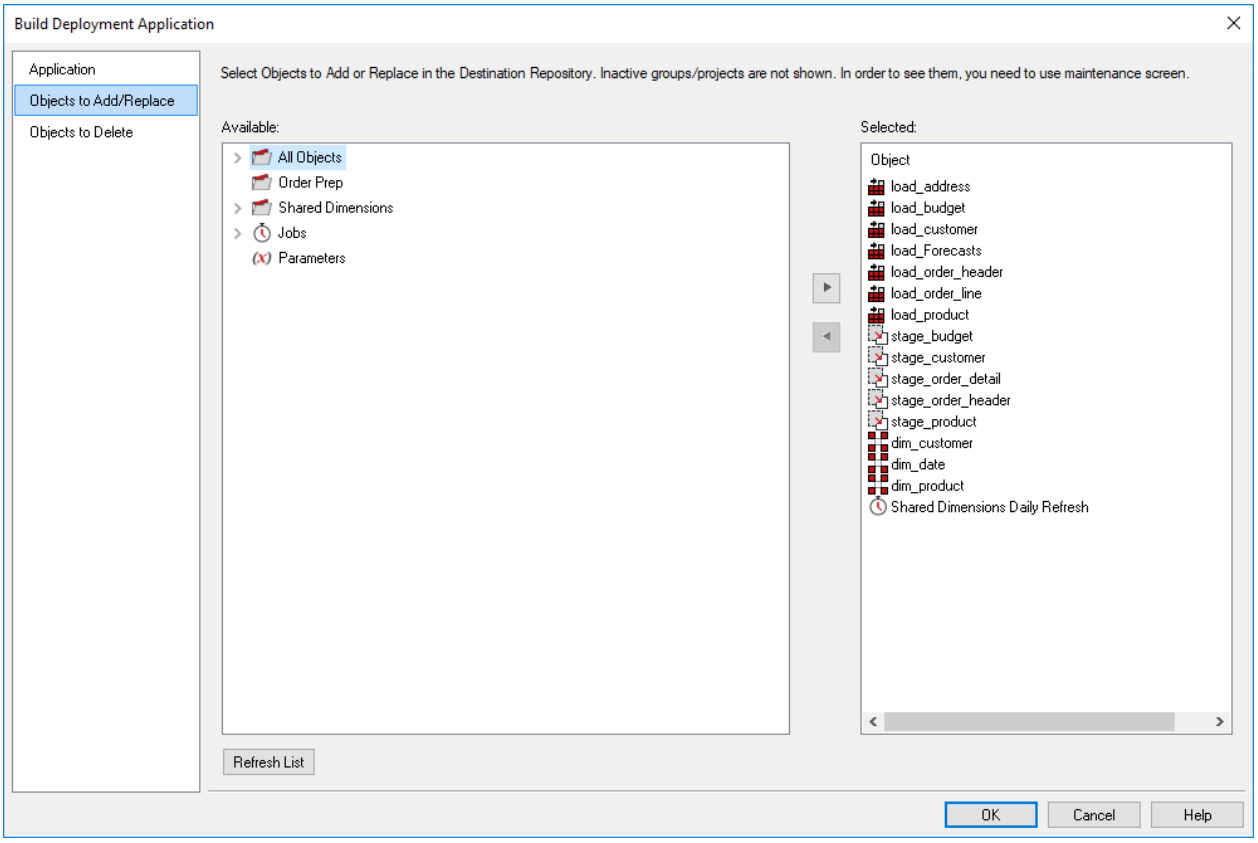


© Copyright, WhereScape Software Ltd All rights reserved.

2. Edit the application details as required.



3. Edit the objects to add or replace as required.

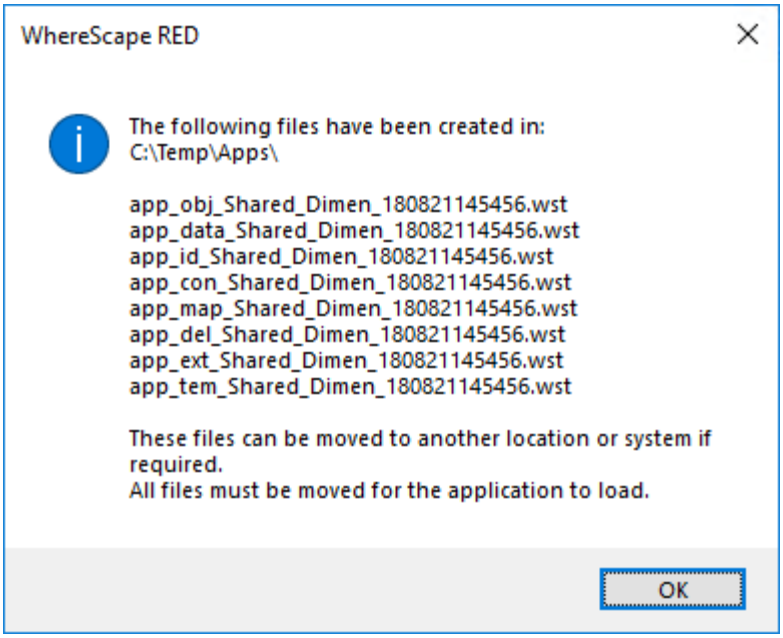


© Copyright, WhereScape Software Ltd All rights reserved.

Note

Creating an application from a job will save the objects in the job and the job, but not the associated objects.

4. Click **OK** when finished.
5. A message window appears, confirming the creation of the application files. Click **OK**.



© Copyright, WhereScape Software Ltd All rights reserved.

Monitoring the Database and Jobs

Note

Oracle on UNIX and Linux only.

WhereScape RED provides a methodology for monitoring both the database and selected jobs. In the event that a job fails or the database is not available a notification message can be sent to an administrator. The monitoring process requires the establishment of an external process. This process must run on a UNIX platform. It does not have to run on the data warehouse database platform, and in fact it is often preferable that it runs on a separate system.

The monitor requires network access to the database. Refer to the RED Installation Guide for information on how to establish the external monitor process. This external monitor process must be running in order to utilize the features discussed in the following sections.

The following sections cover both database monitoring and job monitoring. The normal practice would be to setup database monitoring to cover those times when critical jobs are running. The individual monitoring of critical jobs is then defined.

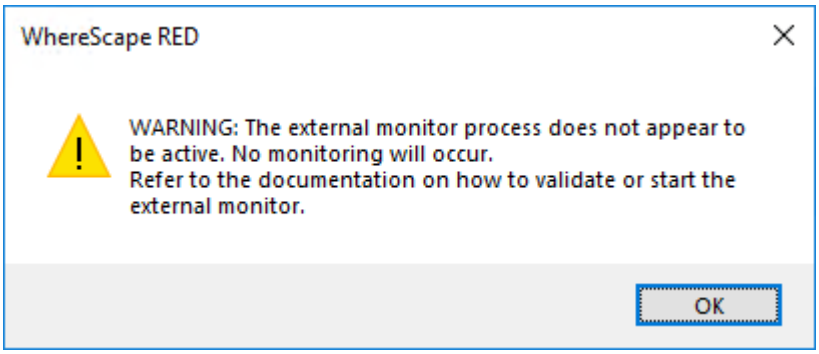
Note

Although it is possible to setup job monitoring without database monitoring this is not recommended. If database monitoring is not in place and the database fails, then no job monitoring will occur and no notifications regarding the jobs will be issued. Therefore, database monitoring is required to provide a comprehensive job monitoring regime.

Database Monitoring

Database monitoring is defined by selecting the **Monitor > Database monitoring** menu option from the Scheduler window, or from the stand-alone scheduler maintenance program.

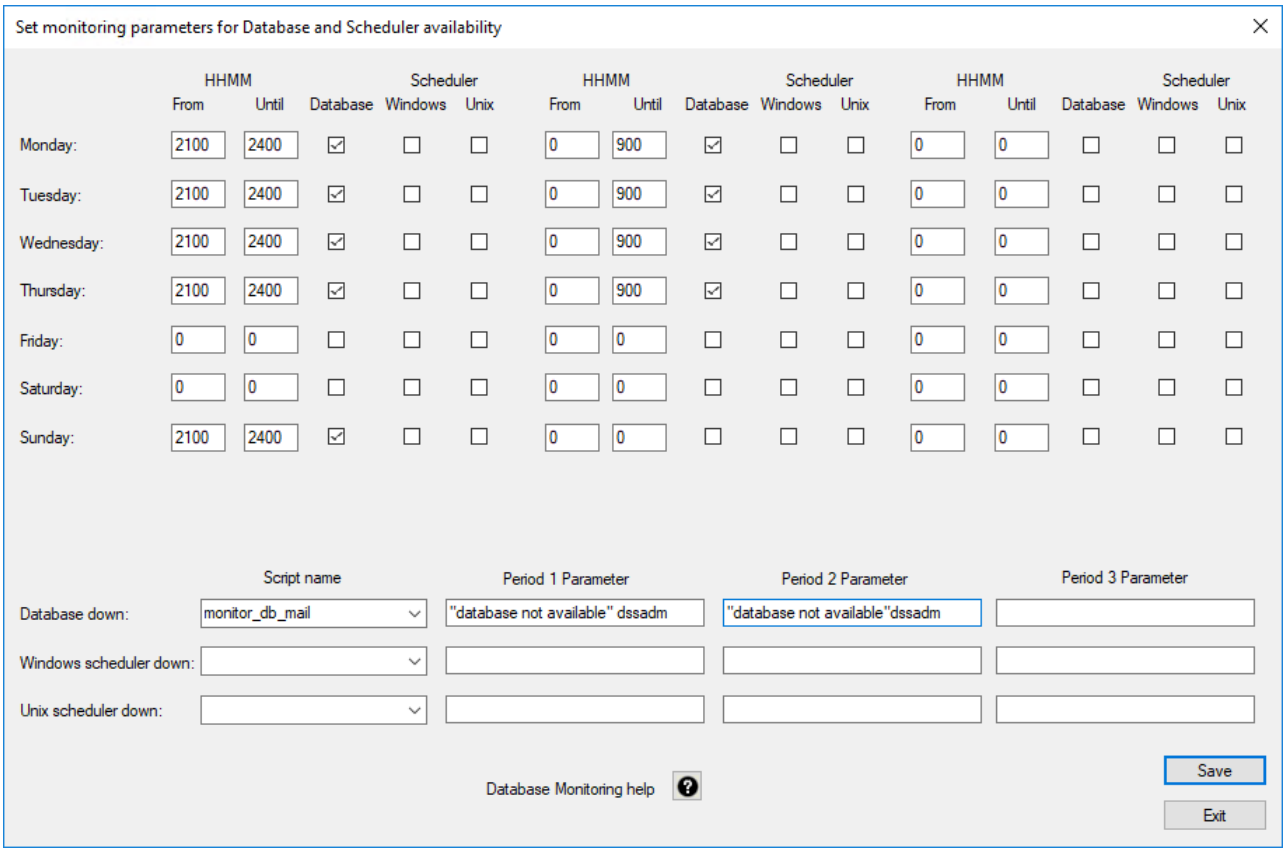
When this menu option is selected, a check is made to ensure that the external monitor process is active. If the external process has not reported in the last 24 hours, the following message appears:



In such a case, the external monitoring process on the UNIX platform needs to be checked or started. Refer to the RED Install guide for instructions on how to maintain the external monitoring process.

© Copyright, WhereScape Software Ltd All rights reserved.

The following window is displayed:



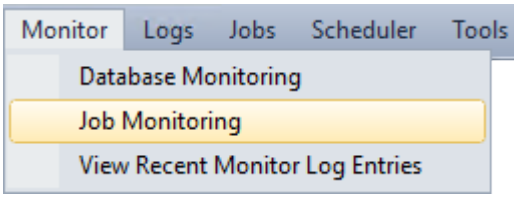
In this example, the database is being monitored between the hours of 9:00pm and 9:00am to cover processing for business days Monday through Friday. If the database is not available during these times, the host script called 'monitor_db_mail' is called and it is passed two positional parameters. The first parameter being "database not available" and the second being "dssadm". The parameters required are specific to the script being invoked.

In this example, we have defined two monitoring periods. The first is for 9:00pm until midnight and the second is from midnight to 9:00am. As a result, there are two sets of parameters defined for a database notification, one for the first period and one for the second period. In this case, both are the same.

Job Monitoring

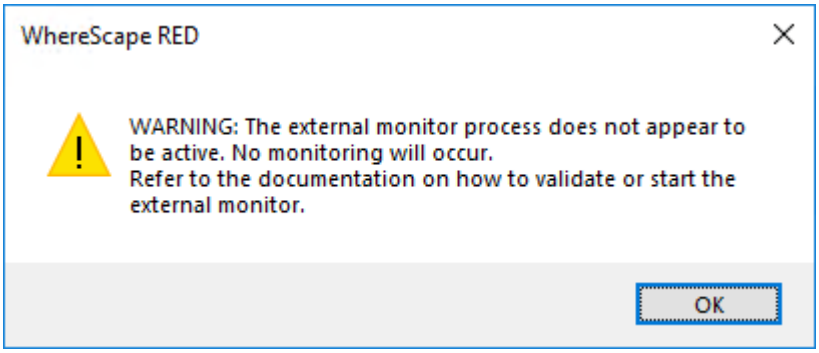
The external monitoring process regularly monitors jobs running in the scheduler and provides notifications when certain conditions occur. By default, the interval between the monitoring of jobs is 15 minutes. This interval can be overridden during the setup of the external job monitor. Refer to the RED Installation Guide for details.

Job monitoring is defined by selecting the **Monitor > Job Monitoring** menu option from the Scheduler window, or from the stand-alone scheduler maintenance program.



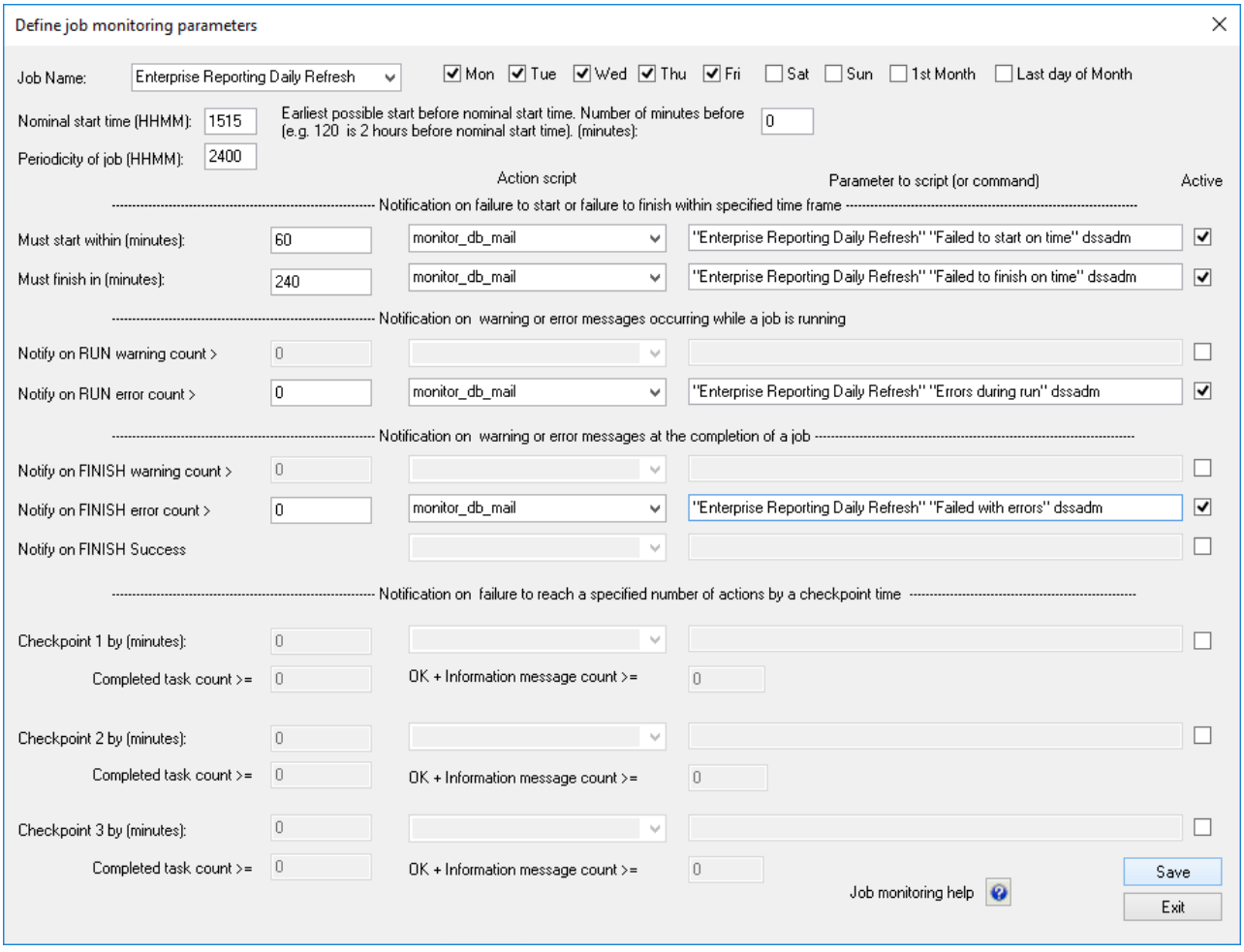
© Copyright, WhereScape Software Ltd All rights reserved.

When this menu option is selected a check is made to ensure that the external monitor process is active. If the external process has not reported in the last 24 hours, the following message appears:



In such a case, the external monitoring process on the UNIX platform needs to be checked or started. Refer to the RED Installation Guide for instructions on how to maintain the external monitoring process.

The following window is displayed:



In this example, the job called 'Enterprise Reporting Daily Refresh' is being monitored. The days Monday through Friday are checked, so monitoring will occur on these days. The nominal start time is 04:00 am. Notifications occur if the following conditions occur.

1. The job fails to start within 60 minutes of this nominal start time.
2. The job fails to finish within 240 minutes (4 hours) of the nominal start time.
3. Errors occur while the job is running
4. The job fails with errors.

© Copyright, WhereScape Software Ltd All rights reserved.

In the above example screen, the user has created an **Action Script** called 'monitor_db_mail' and saved it is a host script object type in RED. This host script is used in all conditions defined above that could cause a notification to occur.

Monitor Days

When defining monitoring for a job, one or more days need to be selected. As well as the normal week days, there is also the first and last day of the month. Using these two special days, it is possible to monitor a job that is critical at the end or start of a month. If Monday through Sunday are checked, then it would not be necessary to set the special days. The monitoring day is the day on which the job starts (or could start).

Note:

The monitoring day is the day of the absolute minimum start time of a job. For example, if we wish to monitor a job that starts at 00:01, and which has an earliest possible start of 5 minutes earlier (e.g. 23:56), then we need to set monitoring for the day of the earliest possible start. Therefore, to monitor the Monday morning run, we must also have the Sunday check box selected.

Nominal Start Time

The nominal start time is the time at which the job will normally start. This field must be populated for all jobs being monitored, even if not used. Monitoring is optimized for daily jobs, so this field is normally set to the start time of the job.

Earliest Possible Start

The earliest possible start field allows for the setting of a possible start time before the nominal start time. In most situations, this field is set to 0. When setting to a value other than zero, the impact of the potential start time on the days being monitored needs to be considered. See the note under the Monitor Days section above. The nominal start time less the earliest possible start time, provides an **Absolute Minimum Start Time** which is used by most of the notification conditions, as well as the monitor days.

Periodicity of Job

The periodicity defines the interval between iterations of the job. For a daily job, this would be 24 hours and 0 minutes or 2400. This periodicity is used by the monitoring software to ascertain if a job being examined is the current iteration of the job or a previous iteration.

Notification Order

Notifications are processed in the following order. If a notification is sent for a job, then no further notification checks are made for that job.

1. Finish Error
2. Finish Warning
3. Finish Success
4. Must Finish In
5. Run Error
6. Run Warning
7. Checkpoint 3
8. Checkpoint 2
9. Checkpoint 1
10. Must Start Within

To fully cover all possible events it is usually necessary to set multiple notifications.

Must Start Within (Notification)

The **Must Start Within** notification allows notification when a job fails to start within a specified number of minutes of the absolute minimum start time. In the example above, the job must start within 60 minutes of the start time.

If this criteria is not met, then the monitoring software will look at the **Action Script** and **Parameter to Script** fields to ascertain how to process the notification. If an action script has been defined, then this script is executed in the UNIX environment and is passed the defined parameters. If a script is not defined, then the **Parameter to Script** field is executed in the UNIX environment.

© Copyright, WhereScape Software Ltd All rights reserved.

Must Finish In (Notification)

The **Must Finish In** notification allows notification when a job fails to finish within a specified number of minutes of the absolute minimum start time. In the example above, the job must finish within 4 hours of the start time. See **Must Start Within** above for the action taken, if the criteria is met.

Run Warning Count (Notification)

The **Run Warning Count** notification allows notification when a job exceeds a specified number of warning messages whilst running. This notification **only occurs when a job is running**. If the job has failed to start, has finished or failed before the external monitor checks the number of warnings this notification will be ignored. See **Must Start Within** above for the action taken, if the criteria is met.

Run Error Count (Notification)

The **Run Error Count** notification allows notification when a job exceeds a specified number of error messages whilst running. This notification **only occurs when a job is running**. If the job has failed to start, has finished or failed before the external monitor checks the number of errors this notification is ignored. See **Must Start Within** above for the action taken, if the criteria is met.

Finish Warning Count (Notification)

The **Finish Warning Count** notification allows notification when a job has exceeded a specified number of warning messages and has completed or failed. This notification **only occurs when a job has completed or failed**. If the job has failed to start or is still running when the external monitor checks the number of warnings, this notification is ignored. See **Must Start Within** above for the action taken, if the criteria is met.

Finish Error Count (Notification)

The **Finish Error Count** notification allows notification when a job has exceeded a specified number of error messages and has failed. This notification **only occurs when a job has failed**. If the job is still running when the external monitor checks the number of warnings, this notification is ignored. If the job has been restarted and gone on to complete normally, this notification is ignored even when errors have occurred. See **Must Start Within** above for the action taken, if the criteria is met.

Finish Success (Notification)

The **Finish Success** notification allows notification when a job completes. This notification can be used for example to mail a log of the completed job to a specified user. See **Must Start Within** above for the action taken, if the criteria is met.

Checkpoint 1 (Notification)

The **Checkpoint 1** notification allows notification when a job fails to achieve either a specified number of task completions, or a specified number of information/OK messages within a specified elapsed time. The elapsed time is the time from the absolute minimum start time. This notification **only occurs when the job is running**. If the job has failed to start, completed or failed when the external monitor checks the checkpoint, this notification is ignored. See **Must Start Within** above for the action taken, if the criteria is not met.

Checkpoint 2 (Notification)

The **Checkpoint 2** notification allows notification when a job fails to achieve either a specified number of task completions, or a specified number of information/OK messages within a specified elapsed time. The elapsed time is the time from the absolute minimum start time. This notification **only occurs when the job is running**. If the job has failed to start, completed or failed when the external monitor checks the checkpoint, this notification is ignored. See **Must Start Within** above for the action taken, if the criteria is not met.

Checkpoint 3 (Notification)

The **Checkpoint 3** notification allows notification when a job fails to achieve either a specified number of task completions, or a specified number of information/OK messages within a specified elapsed time. The elapsed time is the time from the absolute minimum start time. This notification **only occurs when the job is running**. If the job has failed to start, completed or failed when the external monitor checks the checkpoint, this notification is ignored. See **Must Start Within** above for the action taken, if the criteria is not met.

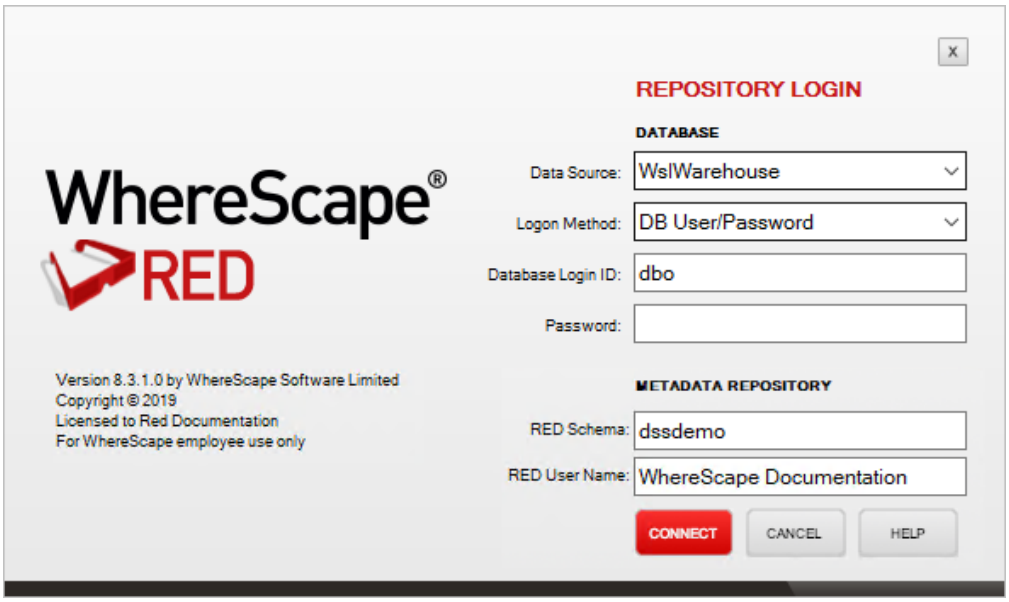
© Copyright, WhereScape Software Ltd All rights reserved.

Stand Alone Scheduler Maintenance

WhereScape RED includes a stand alone scheduler maintenance screen. This screen provides all the scheduler control functionality found in the main WhereScape RED utility, but with no access to the main metadata repository.

Scheduler maintenance logon

The logon screen differs in that the user name and password do not have to be that of a valid metadata repository. This user name/password combination can be any valid database user. There is an additional entry required being **Meta database**. This is the schema of the metadata repository that you wish to connect to in terms of scheduler maintenance.



For **SQL Server** the Meta database must be set to **dbo**.

For **Oracle** data warehouses the 'Meta database' owner must have made the following grants to the specified user before the scheduler maintenance utility can be used.

Scheduler maintenance grants

Repository access grants:

- grant select on ws_meta to dsssched;
- grant select on ws_meta_tables to dsssched;
- grant select on ws_obj_type to dsssched;
- grant select,insert on ws_user_adm to dsssched;
- grant select,insert,update on ws_user_adm to dsssched;
- grant select,alter on ws_wrk_job_dependency_seq to dsssched;
- grant select on ws_table_attributes to dsssched;

Object access (job create) grants:

- grant select on ws_obj_object to dsssched;
- grant select on ws_obj_pro_map to dsssched;
- grant select on ws_obj_project to dsssched;
- grant select on ws_obj_group to dsssched;
- grant select on ws_pro_gro_map to dsssched;

Scheduler status grants:

- grant select on ws_wrk_audit_log to dsssched;

© Copyright, WhereScape Software Ltd All rights reserved.

Scheduler status, and job deletion grants:

- grant select,delete on ws_wrk_error_log to dssched;

Scheduler status, poll grants:

- grant select,update on ws_wrk_scheduler to dssched;

Job creation grants:

- grant select,insert,update,delete on ws_wrk_dependency to dssched;
- grant select,insert,update,delete on ws_wrk_job_ctrl to dssched;
- grant select,alter on ws_job_seq to dssched;
- grant select,alter on ws_task_seq to dssched;

Job maintenance grants:

- grant select,insert,delete on ws_wrk_job_log to dssched;
- grant select,update,delete on ws_wrk_job_run to dssched;
- grant select,insert,update,delete on ws_wrk_dependency to dssched;
- grant select,insert,update,delete on ws_wrk_job_dependency to dssched;
- grant select,delete on ws_wrk_job_thread to dssched;

Task maintenance grants:

- grant select,insert,update,delete on ws_wrk_task_ctrl to dssched;
- grant select,update,delete on ws_wrk_task_run to dssched;
- grant select,insert,delete on ws_wrk_task_log to dssched;
- grant select,insert,update on dss_parameter to dssched;

Right-click Used by option in parameters listing grants:

- grant select on ws_pro_header to dssched;
- grant select on ws_pro_line to dssched;
- grant select on ws_scr_header to dssched;
- grant select on ws_scr_line to dssched;
- grant select on ws_load_tab to dssched;
- grant select on ws_load_col to dssched;
- grant select on ws_stage_tab to dssched;
- grant select on ws_stage_col to dssched;
- grant select on ws_dim_tab to dssched;
- grant select on ws_dim_col to dssched;
- grant select on ws_fact_tab to dssched;
- grant select on ws_fact_col to dssched;
- grant select on ws_agg_tab to dssched;
- grant select on ws_agg_col to dssched;

A sample script to grant these privileges is shipped with WhereScape RED. This script is called 'grant_sched_access.sql' and can be found in the WhereScape program directory.

The scheduler maintenance utility does not require a WhereScape license key. The WhereScape RED software can be installed onto a PC, and use this utility without having to use the WhereScape **Setup Administrator** utility.

© Copyright, WhereScape Software Ltd All rights reserved.

SQL to return Scheduler Status

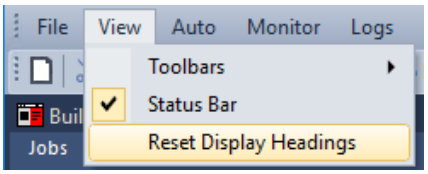
This SQL returns the scheduler status:

```
SELECT CASE
WHEN ws_stop_date IS NOT NULL
THEN 'STOPPED'
WHEN ((DATEDIFF(mi,ws_active_date,GETDATE()) -
CONVERT(INTEGER,DATEDIFF(mi,ws_active_date,GETDATE())/60)*60) > 15 )
OR (CONVERT(INTEGER,DATEDIFF(mi,ws_active_date,GETDATE())/60)>0)
THEN 'NOT ACTIVE'
WHEN (((DATEDIFF(mi,ws_active_date,GETDATE()) -
CONVERT(INTEGER,DATEDIFF(mi,ws_active_date,GETDATE())/60)*60)>((ws_interval/60)+10)
OR CONVERT(INTEGER,DATEDIFF(mi,ws_active_date,GETDATE())/60)>0)
AND ws_poll_flag=1)
THEN 'NOT ACTIVE'
ELSE 'Running'
END
FROM dbo.ws_wrk_scheduler
WHERE ws_name = 'YourSchedulerName'
```

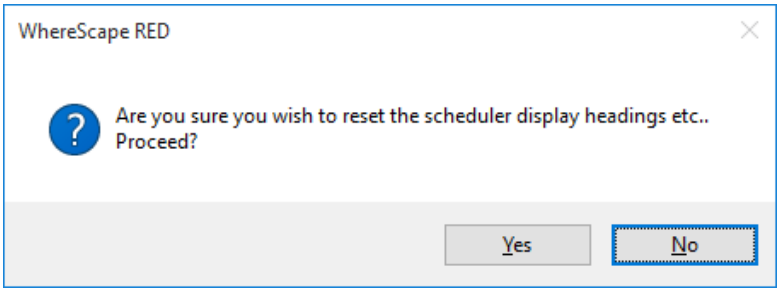
The procedure sets the status in the metadata.

Reset Columns in Job and Task View

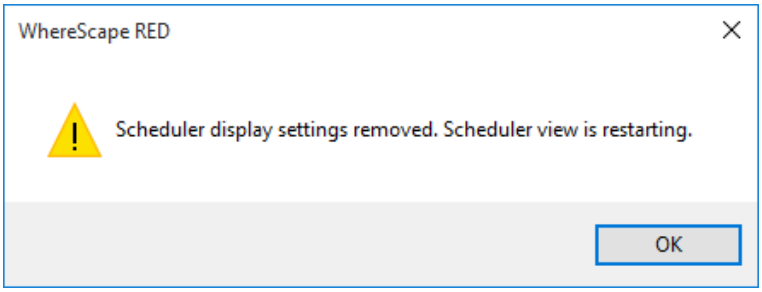
Job and Task Report headings can be reset by selecting the **View > Reset Display Headings** menu option from the Scheduler window. The short-cut keys are Alt+V-R.



A message prompt asks you to confirm the request.



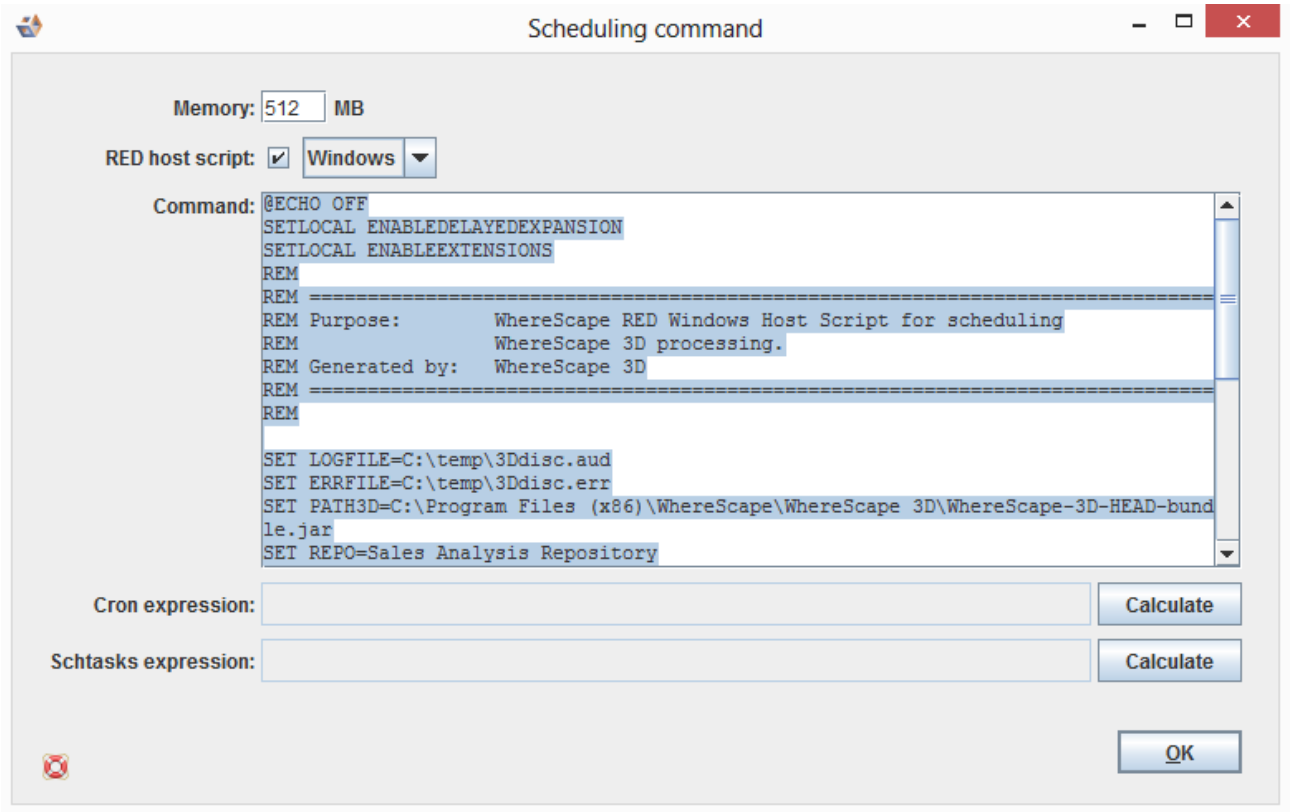
Select **Yes** to reset the display settings, a message is displayed to confirm the reset has occurred.



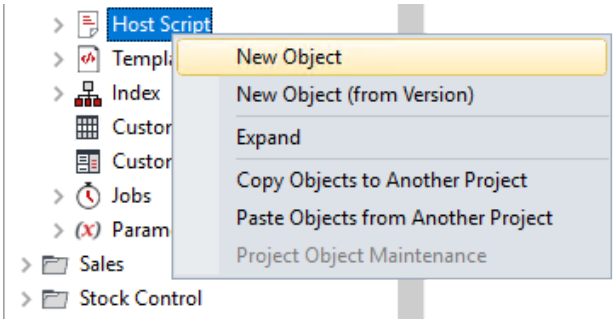
© Copyright, WhereScape Software Ltd All rights reserved.

Scheduling a RED Host Script from 3D

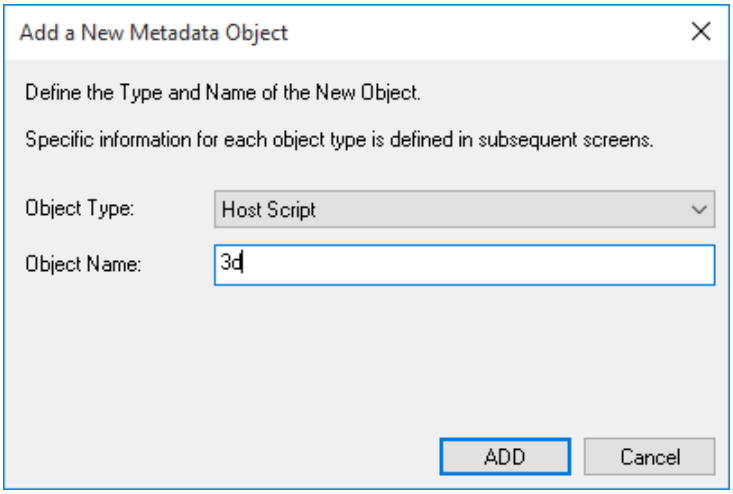
Create the RED host script in 3D. Use Ctrl-C to copy the script.



In RED, create a host script by right-clicking on the **Host Script** object group and choosing **New Object**.

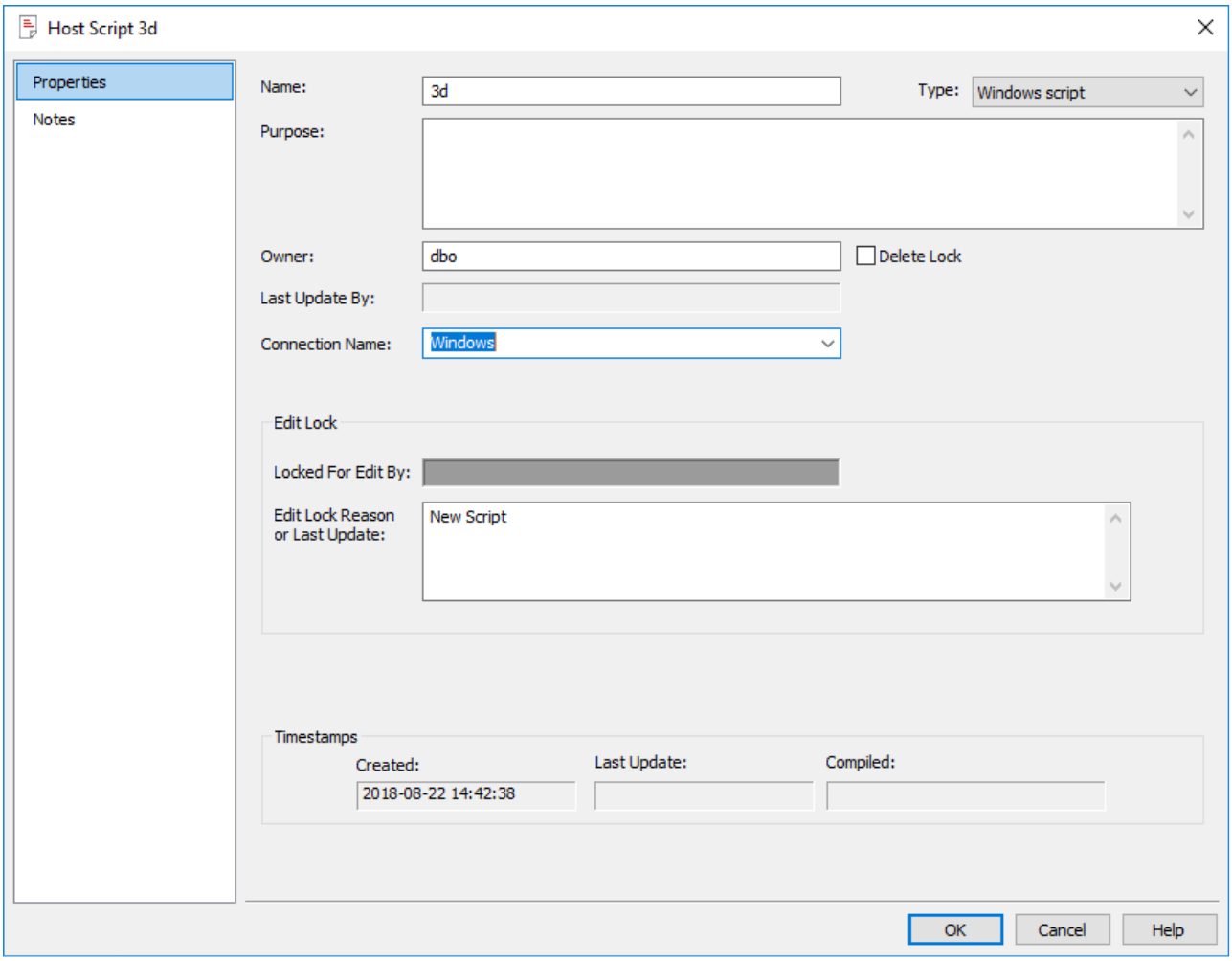


Enter a name for the host script and click **ADD**.



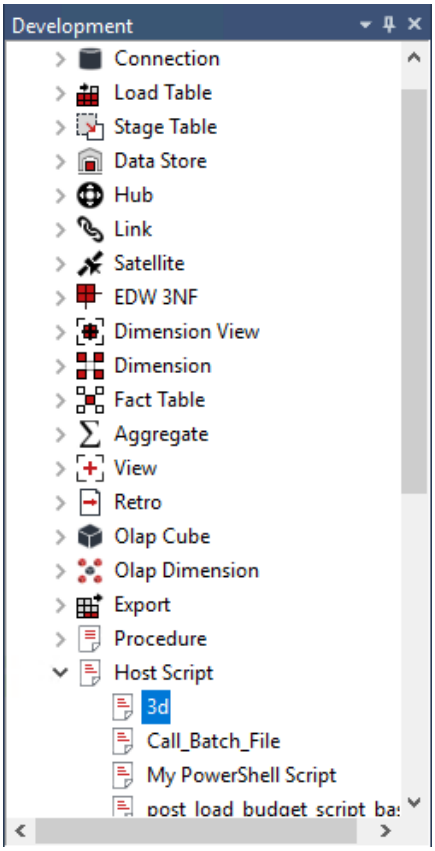
© Copyright, WhereScape Software Ltd All rights reserved.

Set the default connection to Windows and click **OK**.

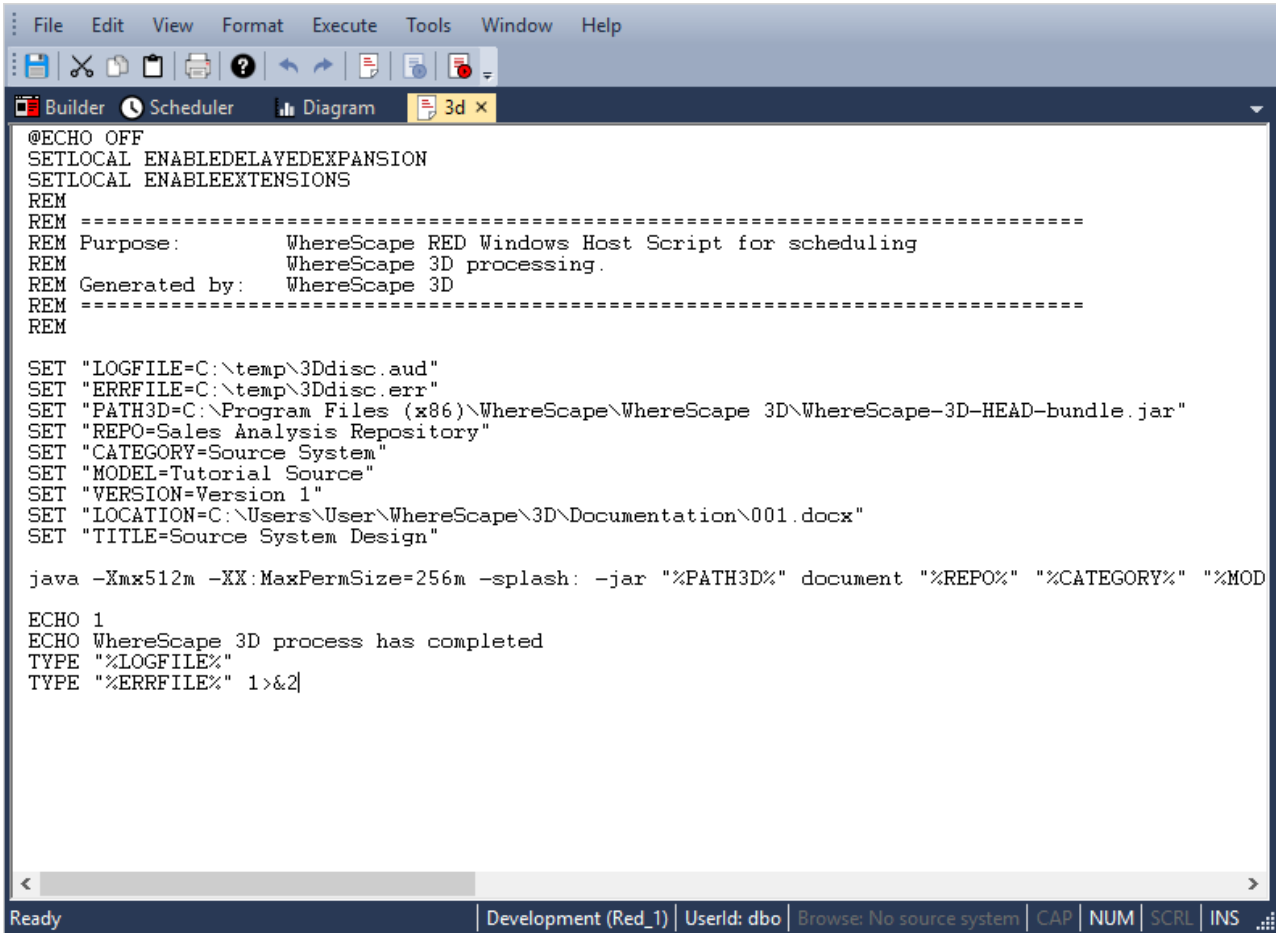


© Copyright, WhereScape Software Ltd All rights reserved.

Double-click the 3d host script in the left pane.

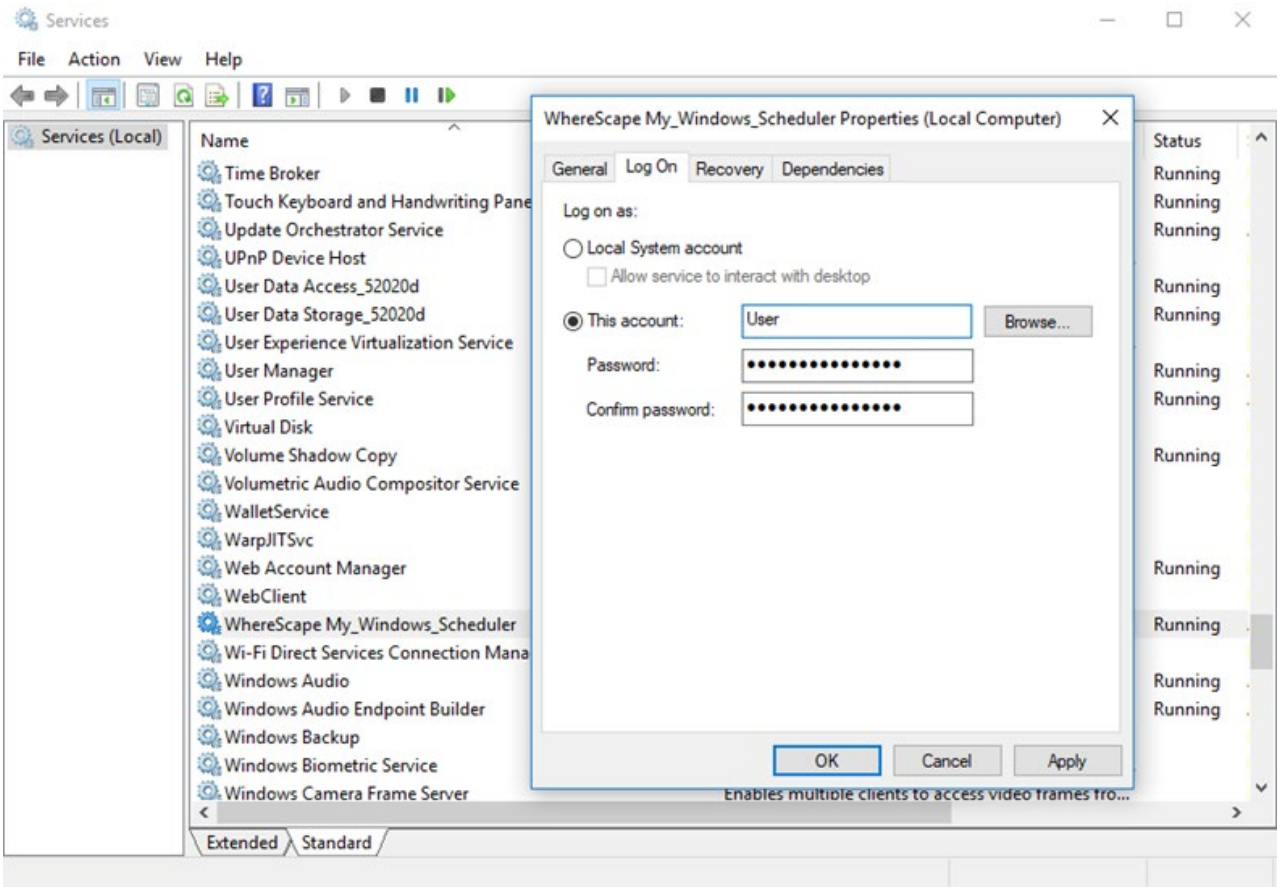


Now paste the script from 3D and click **Save**; then close the host script window.



© Copyright, WhereScape Software Ltd All rights reserved.

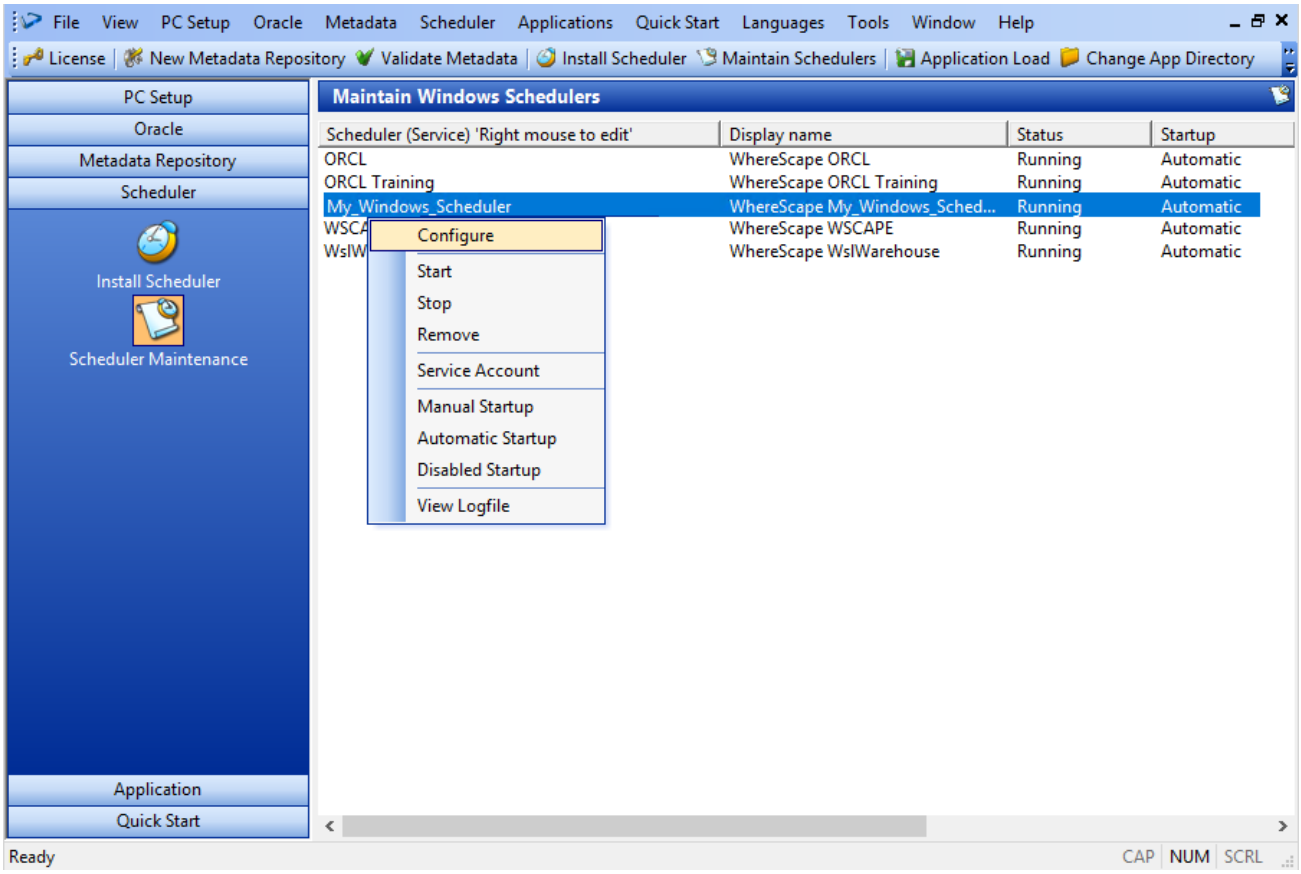
Go to **Control Panel > Administrative Tools > Services** and right-click on the relevant service name. Choose **Properties** and then the **Log On** tab. Change the scheduler to run as the current user. Click **Apply** and then **OK**.



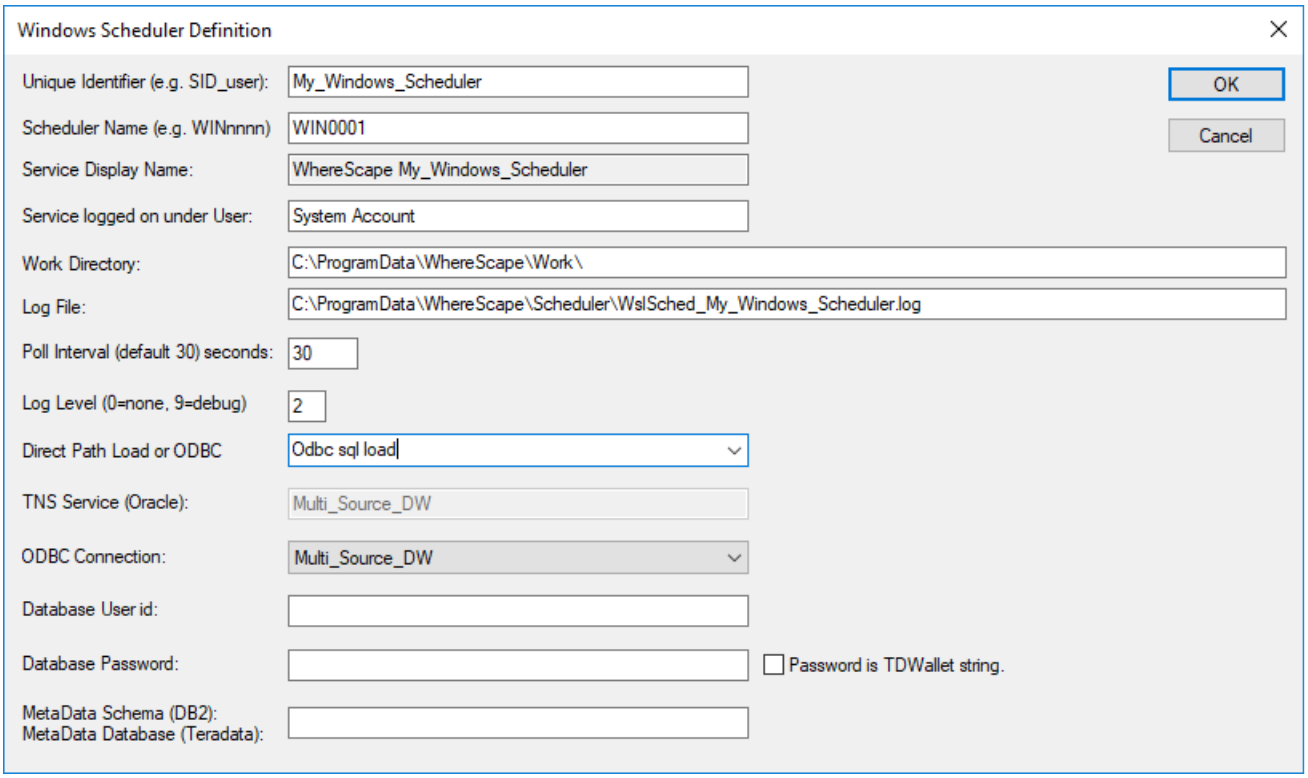
Stop and start the scheduler and then close the **Services** window.

© Copyright, WhereScape Software Ltd All rights reserved.

Click **Scheduler Maintenance** in WhereScape Administrator; right-click the scheduler and choose **Configure**.

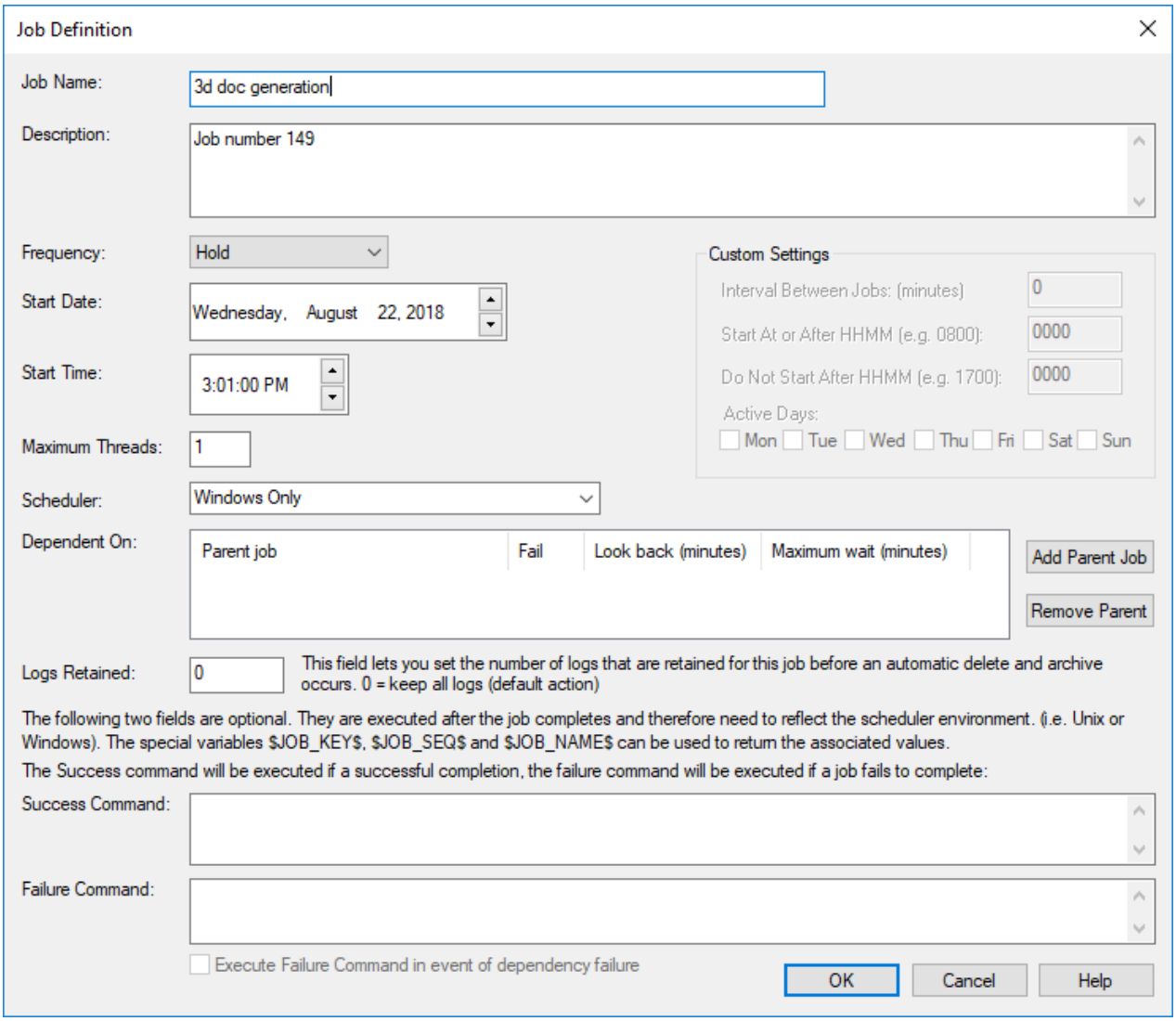


Set the **Service logged on under User** field to the current user. Click **OK** and close WhereScape Administrator.



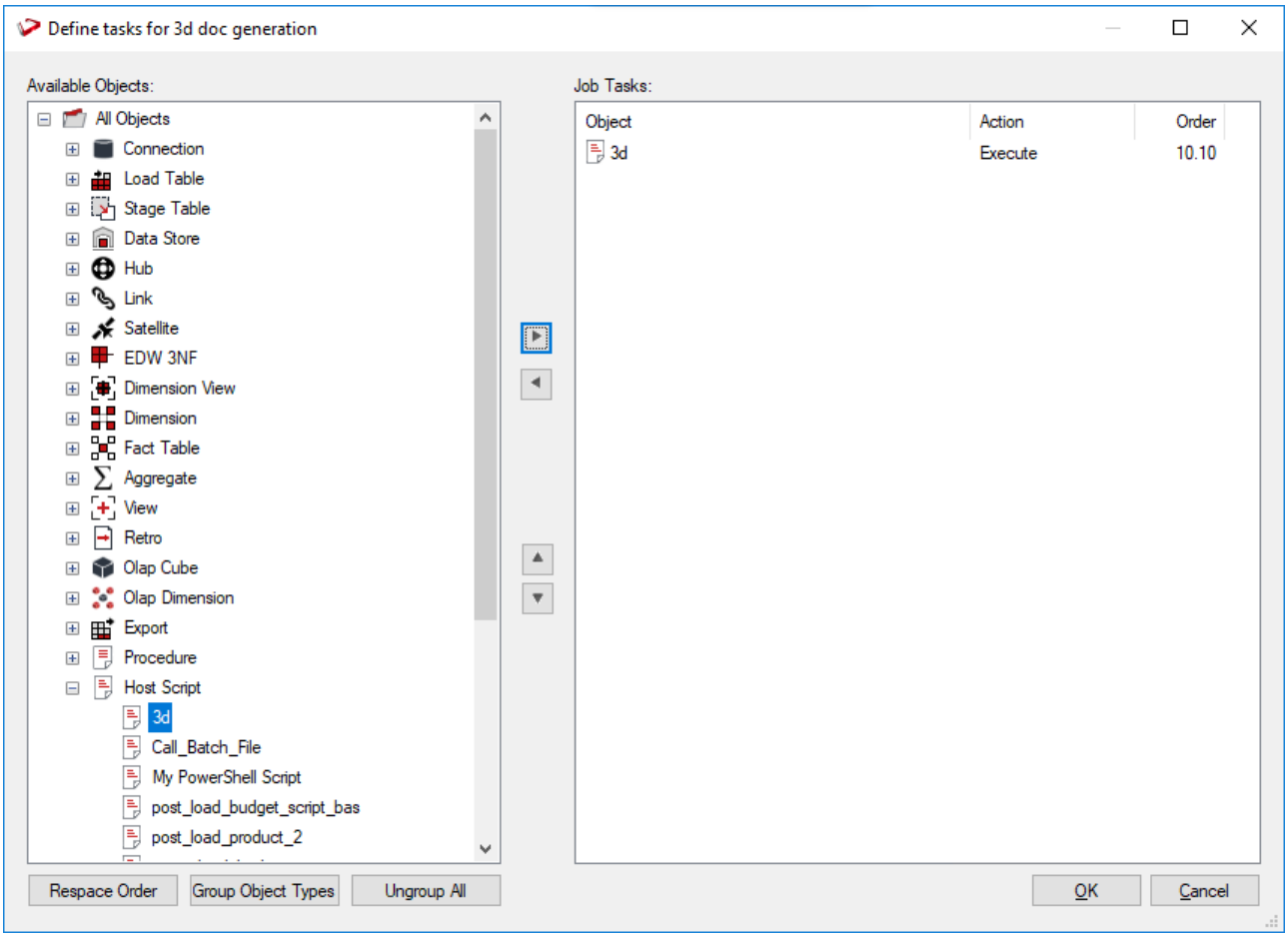
© Copyright, WhereScape Software Ltd All rights reserved.

In RED, click the **Scheduler** tab and create a new job. Click **OK**.



© Copyright, WhereScape Software Ltd All rights reserved.

Select the host script to be run and click **OK**.



Run the job.

© Copyright, WhereScape Software Ltd All rights reserved.

Stopping a Linux/UNIX Scheduler from within RED

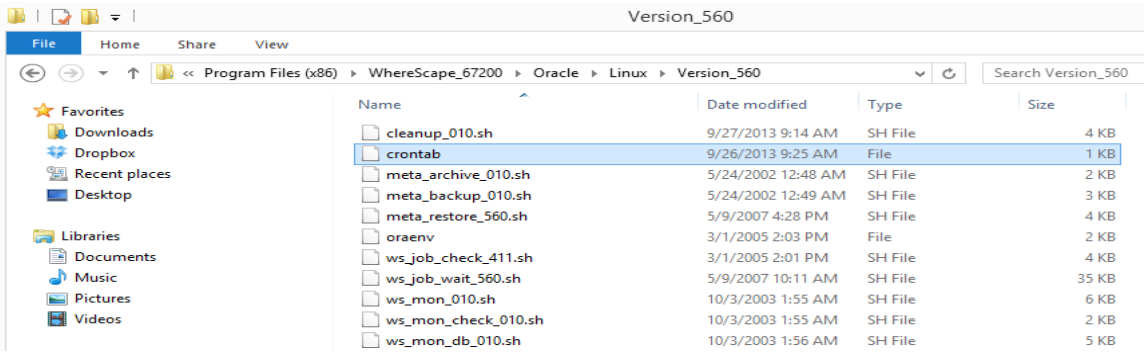
To stop a Linux/UNIX Scheduler from within RED, follow the steps below:

1. Edit the **crontab** and comment out the ws_sched_check_nnn.sh entry. This will stop the scheduler restarting within the next 20 minutes.

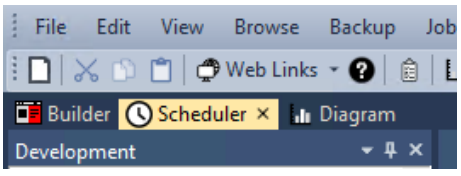
```
## backup of the meta data
## workdays backup at end of day
## on the weekends at 7:00pm
##
00 20 * * 1-5 _HOME_/wsl/bin/meta_backup_010.sh
oraenv >/dev/null 2>&1
00 19 * * 0,6 _HOME_/wsl/bin/meta_backup_010.sh
oraenv >/dev/null 2>&1
##
## each day archive (tar) the meta backups and compress
##
00 22 * * * _HOME_/wsl/bin/meta_archive_010.sh >/dev/null 2>&1
##
## each day cleanup log files etc.
##
00 18 * * * _HOME_/wsl/bin/cleanup_010.sh oraenv >/dev/null 2>&1
#
## check and if required start the scheduler
##
0,20,40 * * * * _HOME_/wsl/bin/ws_sched_check_560.sh
oraenv >>_HOME_/wsl/sched/log/sched_oraenv.log 2>&1
#
## check and if required start the monitor
##
0,30 * * * * _HOME_/wsl/bin/ws_mon_check_010.sh
oraenv >>_HOME_/wsl/sched/log/mon_oraenv.log 2>&1
```

Note

There may be several different versions of the scheduler files for a given database and platform (UNIX or Linux). For example, there may be different folders in ...\WhereScape\Oracle\Linux\: Version_010, Version_411 and Version_560. The highest version number script less than or equal to the version of RED in use should always be used.

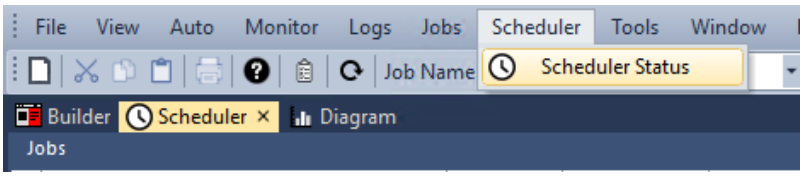


2. Start RED and click the **Scheduler** tab.

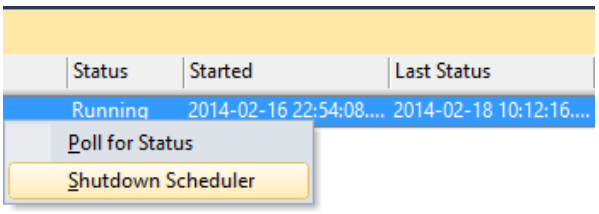


© Copyright, WhereScape Software Ltd All rights reserved.

3. Click the **Scheduler** menu from the toolbar and then select **Scheduler Status**.



4. Right-click the displayed UNIX/Linux scheduler entry and choose **Shutdown Scheduler**.



Sometime within the next poll interval of the scheduler, the scheduler will gracefully stop.

© Copyright, WhereScape Software Ltd All rights reserved.

Indexes

Indexes may exist on any table defined in WhereScape RED.

By default, WhereScape RED will auto-generate a number of indexes during the drag and drop process to create new table objects and when building procedures.

The **Enable Automatic Creation of Indexes** option in the **Connection Properties** window enables you to turn ON/OFF the automatic creation of metadata for indexes when creating objects. Neither the RED metadata nor the physical index is created, if this setting is not enabled.

The generated indexes can be altered or deleted. New indexes can be created as desired.

Note

The maintenance of the indexes is performed as part of the normal scheduler processing.

In the left pane, right-click on a table to:

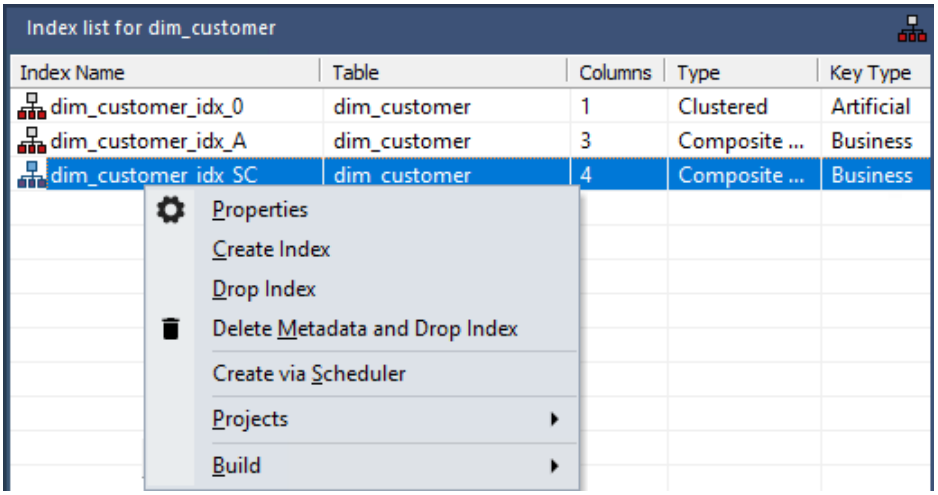
- Display indexes
- Add indexes

Index Definition

Right-clicking a table in the left pane and selecting **Display Indexes** from the context menu lists the indexes for the table in the middle pane. Alternatively, you can double-click the Index object type in the left pane, to display all indexes in the repository or a specific group or project.

In the middle pane, right-click an index and the following options are displayed:

- Properties
- Create Index
- Drop Index
- Delete Metadata and Drop Index
- Create via Scheduler
- Projects
- Build

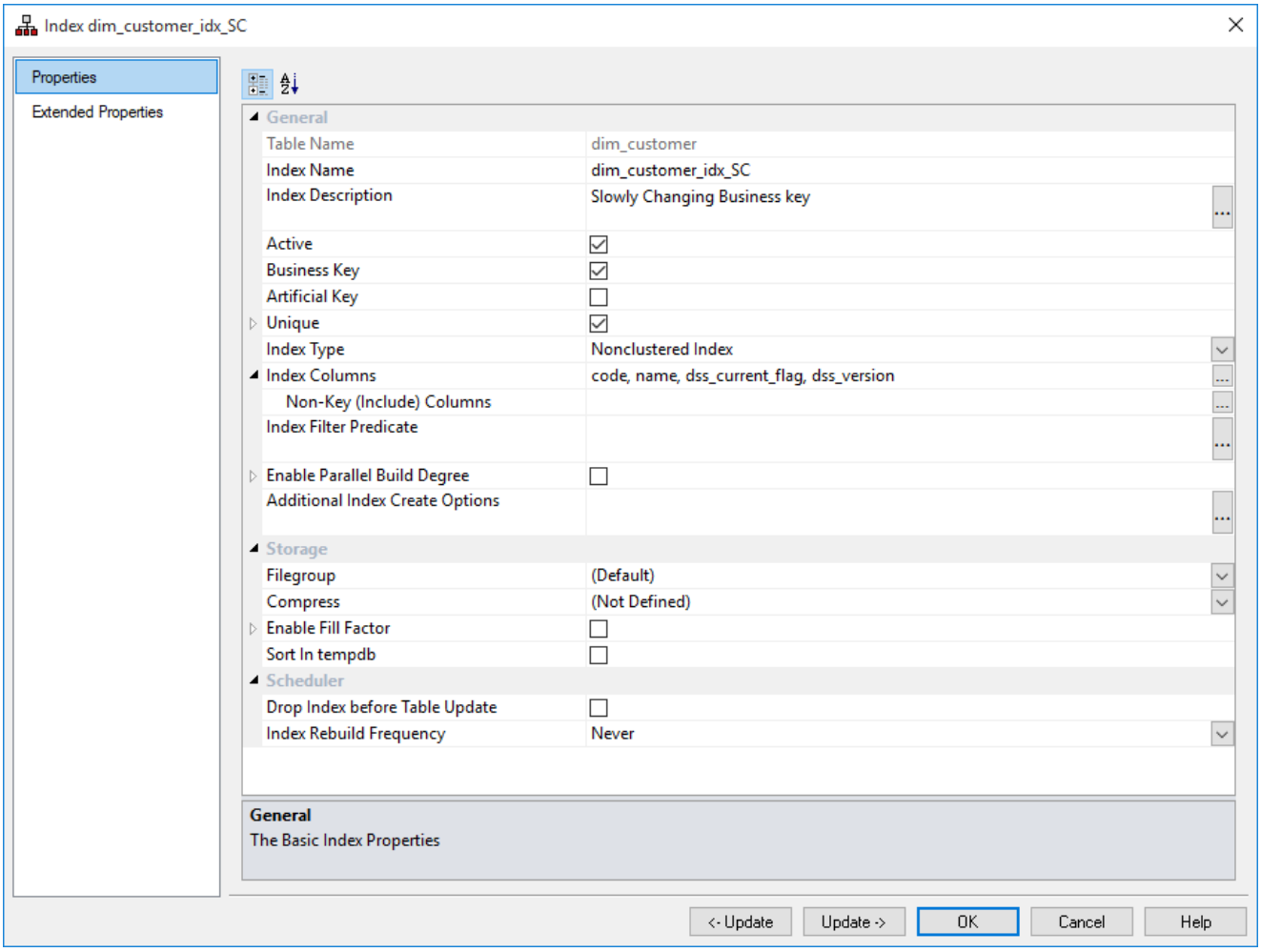


© Copyright, WhereScape Software Ltd All rights reserved.

Properties

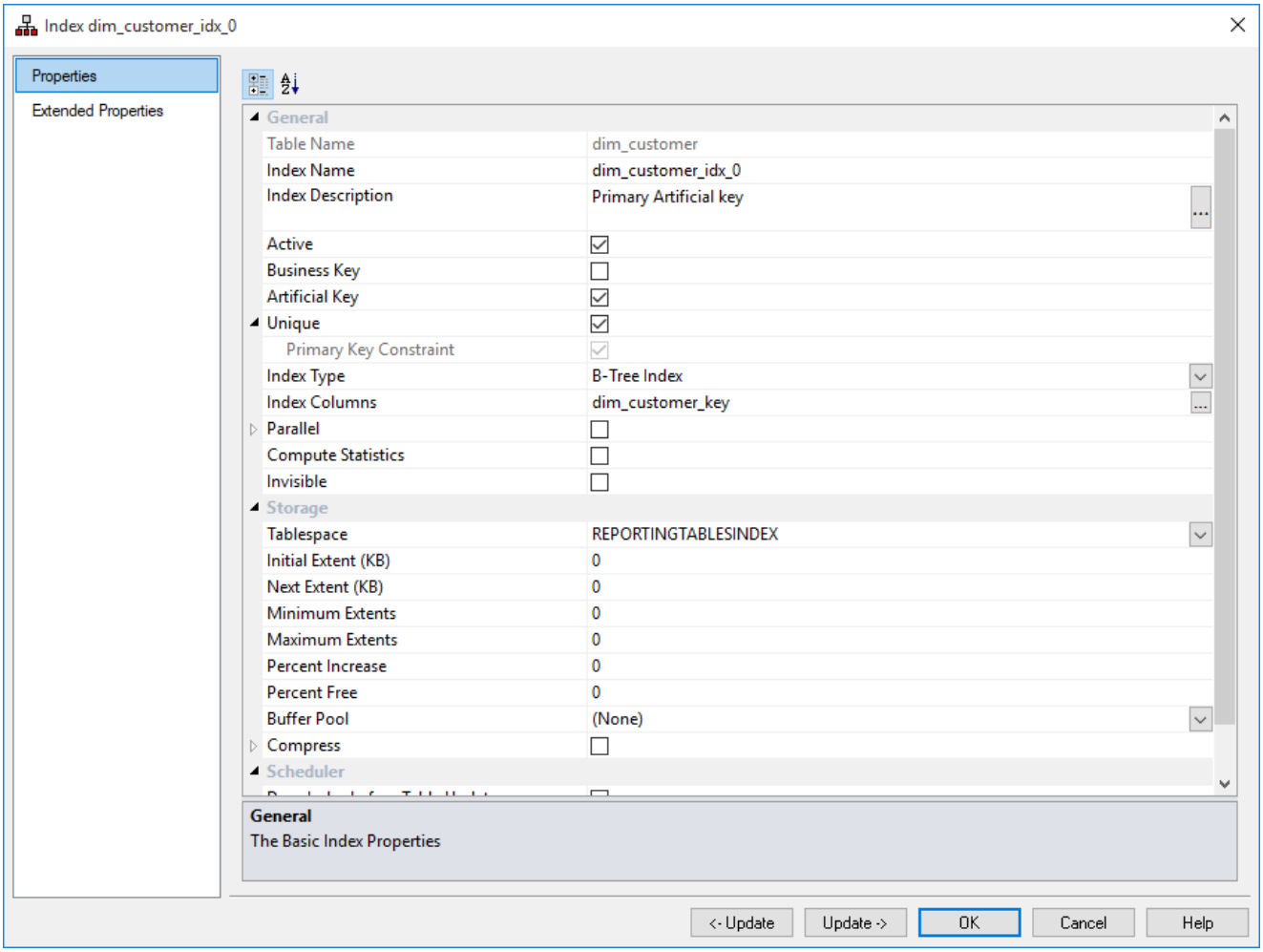
The **Properties** screen (see example below) can be selected via the right-click context menu when positioned on an index name in the middle pane. The Update Buttons: **Update <-** and **Update ->** are used to move to the previous and next index respectively. The Update Buttons are not available when browsing all indexes in a group, project or repository.

SQL Server example:



© Copyright, WhereScape Software Ltd All rights reserved.

Oracle example:

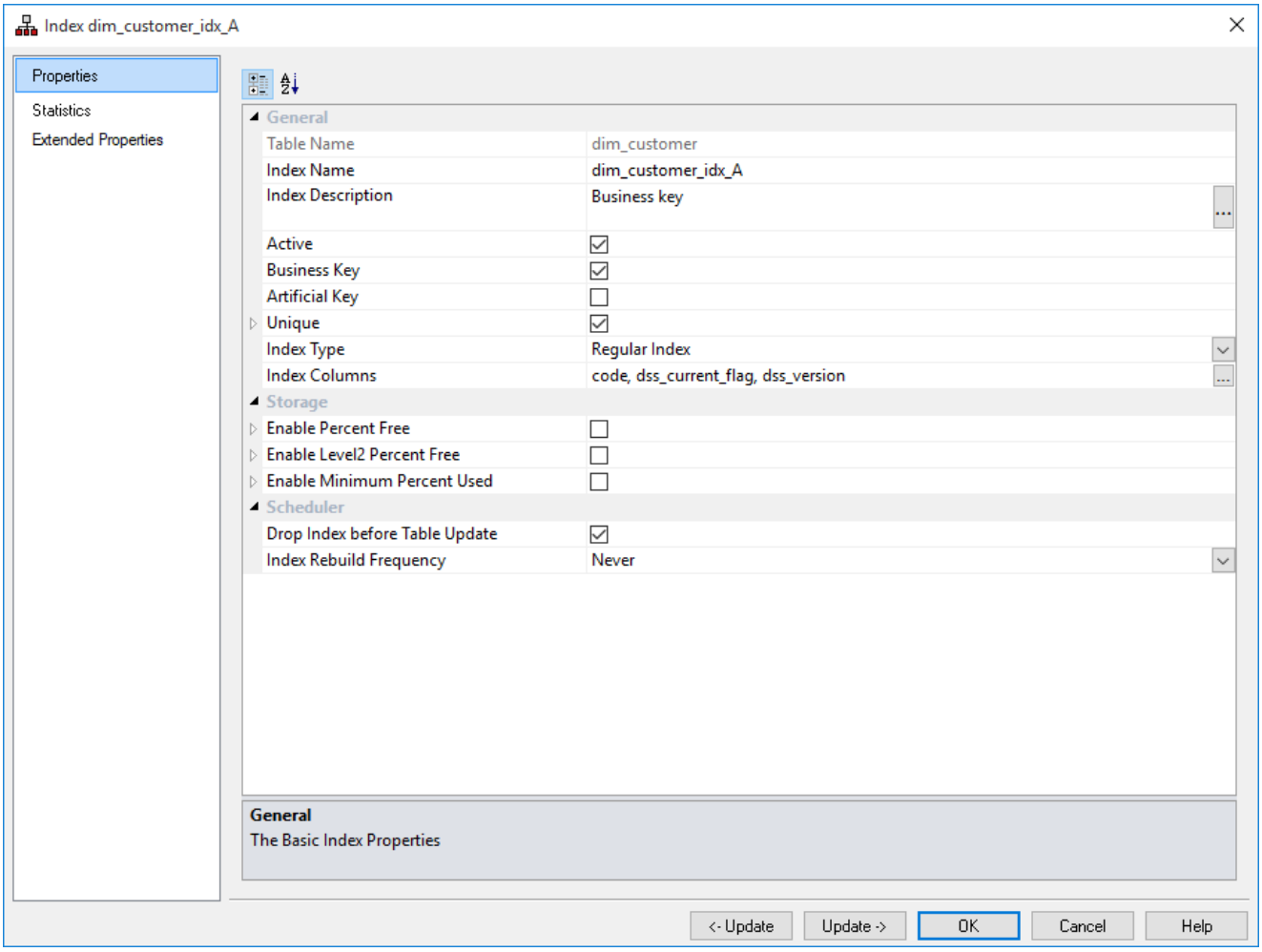


Note

- Oracle 11g allows Indexes to be marked as **invisible**.
- **Invisible Indexes** are maintained like any other index but they are ignored by the optimizer, unless the OPTIMIZER_USE_INVISIBLE_INDEXES parameter is set to true at the instance or the session level. The default of this option is not set.

© Copyright, WhereScape Software Ltd All rights reserved.

IBM **DB2** example:



The fields in the **General** section are described below:

Field	Description
Index Name	Typically the table short name followed by: • **_idx_0** indicating primary key • **_idx_n** where n = any number from 1 indicating dimensional keys • **_idx_x** where x = any letter a thru z indicating business keys.
Index Description	Free flow description of the index.
Active	• When enabled means the index is in use. • When disabled means the index is not managed by the scheduler.
Business Key	Denotes a business key.
Artificial Key	When enabled indicates that this is the surrogate (artificial) key generated by the system.
Unique	Specifies that the index is a unique index. **Note** If both **unique and artificial** are set it is assumed to be a primary key constraint and it is added as such.
Index Type	Database-specific type of index On **SQL Server** the options are: • Nonclustered Index • Clustered Index

© Copyright, WhereScape Software Ltd All rights reserved.

Field	Description
	• ColumnStore Index available for SQL2012 On **Oracle** the options are: • B-Tree Index • Bitmap Index • Function-based Index
Index Columns	Shows the columns in the order they will be applied to the index. The order can be changed using the up/down buttons at the left.
Non-Key (Include) Columns	Extend a Non-clustered Index to include Non-Key columns in addition to the key columns.
Index Filter Predicate	A filtered index is an optimized Non-clustered index, which uses a filter predicate to
Enable Parallel Build Degree	Activates parallel creation /build of the index. For unsupported database versions (Pre SQL2005) leave disabled.
Additional Index Create Options	Database-specific-and-compliant options to include in the generated CREATE INDEX statement.
Compute Statistics	**For Oracle**. When enabled, RED automatically collects statistics during index creation and rebuild. NOTE: Deprecated since 10g (but supported for backwards compatibility) as now statistics are always collected during index creation and rebuild.

The fields in the **Storage** section are described below:

Field	Description
Filegroup/Tablespace	Default Filegroup/Tablespace of the index that determines the storage location it is created in. Select (Default) to use the default.
Initial Extent (KB)	**For Oracle**. Size of the first extent allocated to the index when it is created. The default value of 0 will instead use the value from the Tablespace in which the index is stored.
Next Extent (KB)	**For Oracle**. Size of the next extent allocated to the index, unless Percent Increase is specified.
Minimum Extents	**For Oracle**. Number of extents to allocate when the index is created. The default value of 0 will instead use the value from the Tablespace in which the index is stored.
Maximum Extents	**For Oracle**. Total number of extents that can be allocated to the index. The default value of 0 will instead use the value from the Tablespace in which the index is stored.
Percent Increase	**For Oracle**. Percent by which the third and subsequent grow over the preceding extent. The default value of 0 will instead use the value from the Tablespace in which the index is stored.
Percent Free	**For Oracle**. Percentage [0-99] of space in each data block of the index to reserve for future updates. The default value of 0 will instead use the value from the Tablespace in which the index is stored.
Buffer Pool	**For Oracle**. Default buffer pool used to cache the data blocks of the index.
Compress	Compress index to reduce disk and memory use. For unsupported database versions (Pre SQL2008) select (Not Defined).
Enable Fill Factor	Setting this will enable setting of the Fill Factor. For unsupported database versions (Pre SQL2005) leave disabled.
Sort In tempdb	Use the tempdb to store the intermediate sort results that are used to build the index, which may reduce the index build time if tempdb is on different disks to the index.

© Copyright, WhereScape Software Ltd All rights reserved.

Field	Description
Enable Percent Free	**For DB2.** Setting this will enable setting of the Percent Free to something other than the Database defaults.
Percent Free	**For DB2.** The amount of free space to leave in each extent/page of the index.
Enable Level2 Percent Free	**For DB2.** Setting this will enable setting of the Level2 Percent Free to something other than the Database defaults.
Level2 Percent Free	**For DB2.** The amount of free space in each index level 2 page.
Enable Minimum Percent Used	**For DB2.** Setting this will enable setting of the Minimum Percent Used to something other than the Database defaults.
Minimum Percent Used	**For DB2.** The threshold of the minimum percentage of space used on an index leaf page.

The fields in the **Scheduler** section are described below:

Field	Description
Drop Index before Table Update	Drop the index before running the table update procedure and recreate the index after the update procedure has completed.
Index Rebuild Frequency	Optional frequency that the index is automatically rebuilt by the Scheduler.

Indexes are normally managed by the scheduler as part of the normal processing of a table.

Refer to **Ws_Maintain_Indexes**. This function allows the control of index drop and creation from within a procedure. Typically this function is called when using partitioned tables.

Extended Properties

Extended property variables can be set for index objects. An extended property's scope is defined by object type but the value for the variable can be set for the table and each index independently. Refer to **Creating an Extended Property Definition** for details.

Automatically created indexes are recreated each time the table's procedure is rebuilt. Therefore, the values set for extended properties in automatically created indexes are lost when they are recreated. Manually created indexes and their extended properties are persisted.

As with extended properties defined for tables and connections, extended properties defined in indexes can be accessed from templates, including create index DDL, drop index DDL and update procedure/script.

© Copyright, WhereScape Software Ltd All rights reserved.

Documentation and Diagrams

WhereScape RED includes the ability to document the data warehouse based on the information stored against the metadata for all the tables and columns in the data warehouse.

The documentation will only be meaningful if information is stored in the metadata. The business definition and a description should be stored against all columns that will be visible to end users.

The following sections describe how to **generate** and **read** the documentation.

Also included is a section on information that is available to assist in the connection of end user **query tools**.

Creating Documentation

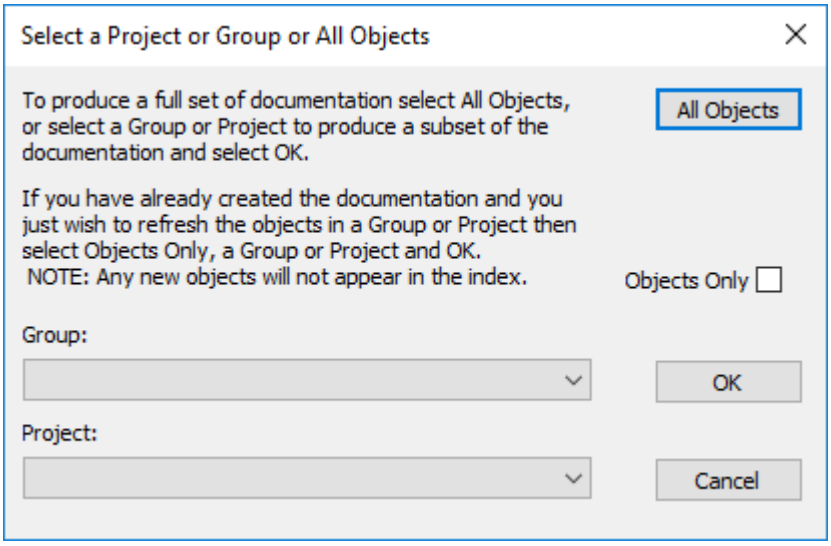
Create documentation

To create the documentation for the components of the data warehouse, select **Doc** from the builder menu bar, then **Create Documentation**.

If the repository has projects or groups in use, then the following dialog will appear to allow the selection of a specific group or project. The default is all objects in the repository.

Note

Projects or groups that have been flagged as inactive are not displayed/available from the Group and Project drop-down lists. Refer to **Maintaining Group/Project Active Flag** for details.

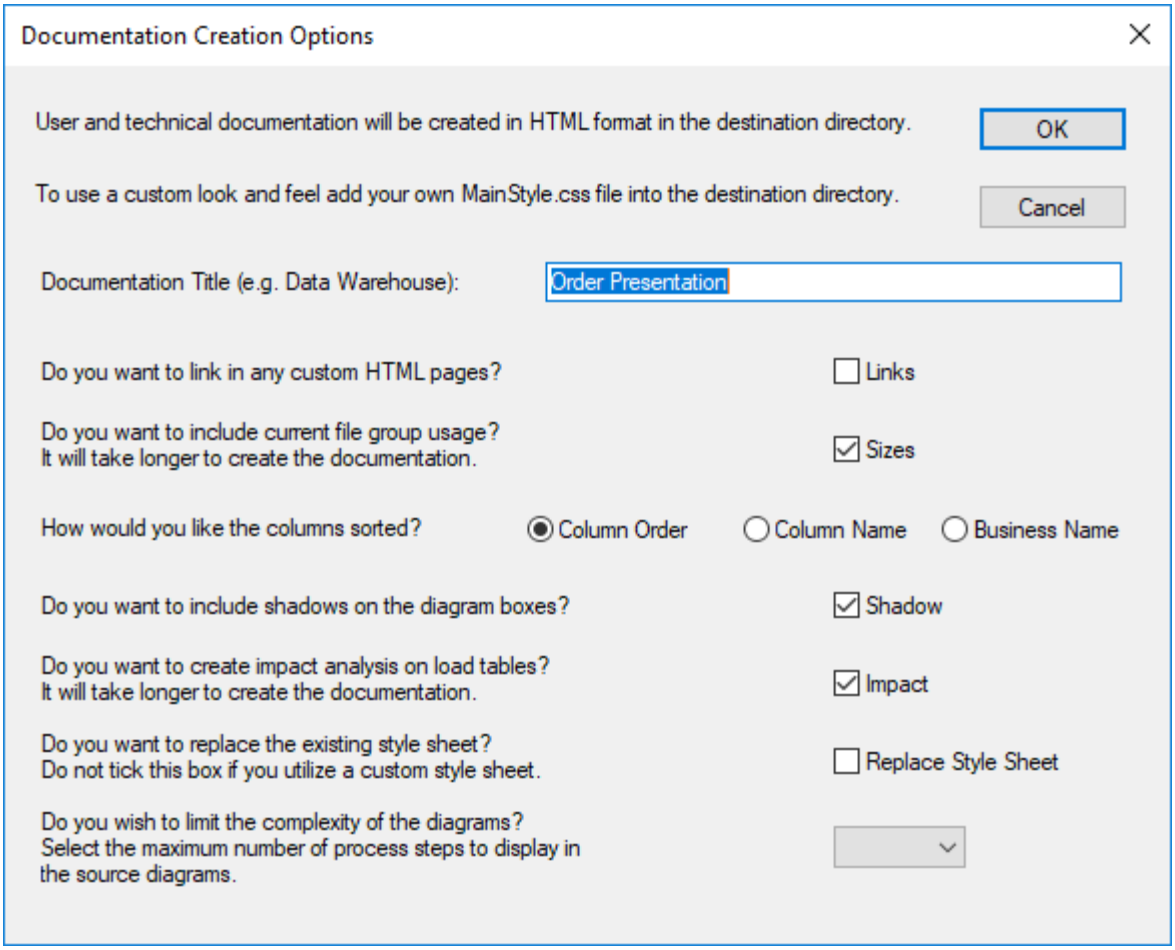


A file dialog appears next. Select a file path (directory) where generated HTML files are saved.

A style sheet called **mainstyle.css** is created if it does not exist. If this style sheet exists then it is not overwritten. The style sheet can therefore be modified to match an existing intranet standard.

© Copyright, WhereScape Software Ltd All rights reserved.

The next screen enables the inclusion of a banner and user defined links. It also provides some build options:



The size checkbox instructs the documentation generator to examine the size of all tables and indexes in the database. This process may be slow in some situations, thus should normally be used only for final documentation.

The sorted checkbox sorts the columns within the tables into alphabetical order. By default the columns are in the order in which they appear within the tables.

Creating a header

If you check the banner frame option then a banner (heading) will be created at the top of each page in the documentation. You will be prompted for height of the banner frame in pixels (default is 60), an image file (jpg or gif) and any banner text. It is recommended that any image be relatively small (60 pixels high or approximately 1/2 an inch) as it will appear on every page.

Adding Links

Custom information can be linked into the generated documentation.

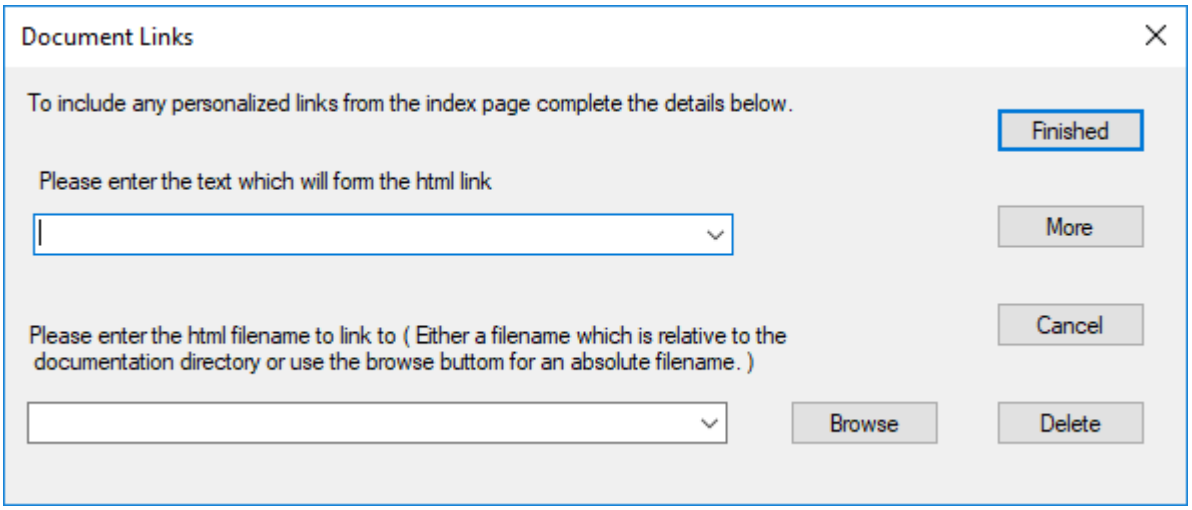
This means that every time the documentation is regenerated, custom information will be included. In this way the complete documentation for the data warehouse can be viewed in one location.

If you select the add links option then you are prompted to include personalized links from index pages. These links must be to previously created HTML files.

Index pages (linkage points) are available at three points:

- index - initial page
- techindex - technical documentation initial page
- indexuser - user documentation initial page

© Copyright, WhereScape Software Ltd All rights reserved.



Multiple links can be added to each index page by using the **More** option.

Adding glossary elements

As part of the user documentation a glossary is produced defining all the business terms and column names used in the data warehouse. This glossary is primarily based on the columns in the dimension and fact tables. Additional information can, however, be added via the 'Ws_Api_Glossary' procedure defined in the procedures chapter. This procedure allows the manual inclusion of glossary elements that will be stored in the metadata repository and added to the glossary whenever the documentation is recreated.

Batch Documentation Creation

WhereScape RED includes the ability to document the data warehouse, based on the information stored against the metadata for all the tables and columns in the data warehouse. In a larger environment, it may be a good idea to generate documentation in batch mode.

The following syntax chart illustrates the options available:

```
med.exe /BD { /U UserName { /P Password } } /C OdbcSource { /M Schema } { /N
FullName } /D Directory { /G GroupName | ProjectName } /S NumHops { /I- }
```

Note:

{ } indicates an optional parameter and | indicates alternative values.

Parameter Descriptions

The following parameters are available:

Parameter	Specify Value?	Mandatory?	Description
BD	No	Yes	Indicates batch documentation mode.
U	Yes	Sometimes *1	ODBC username parameter.
P	Yes	Sometimes *1	ODBC password parameter.
C	Yes	Yes	ODBC data source parameter.
A	Yes	No	ODBC DSN architecture.
M	Yes	Sometimes *2	Metadata database parameter.
N	Yes	No	RED user name parameter, only for logging purposes.
D	Yes	Yes	Directory name where documentation is created.

© Copyright, WhereScape Software Ltd All rights reserved.

Parameter	Specify Value?	Mandatory?	Description
G	Yes	No	Group or Project name if specified. All Objects if not included.
S	Yes	Yes	Number of processes/hops in the source diagrams.
I-	No	No	Exclude Impact Analysis on load tables (reduces the time required to generate documentation)

Notes

- *1. User Name and Password are not required when using a trusted connection or operating system authentication.
- *2. Metadata Schema is required for DB2 but is not required for Oracle and SQL Server.

Example

The following example connects to a SQL Server repository called WslWarehouse, using a trusted connection into the C:\Temp\my_doco directory with 4 hops in diagrams:

```
med /BD /C "WslWarehouse" /D "C:\Temp\my_doco" /S "4"
```

Reading the Documentation

To read the documentation you have created, select **Doc** from the builder menu bar, then **Display Documentation**. This will launch a browser and display the contents of index.html. Alternatively, you can access the HTML pages directly from their saved location.

© Copyright, WhereScape Software Ltd All rights reserved.

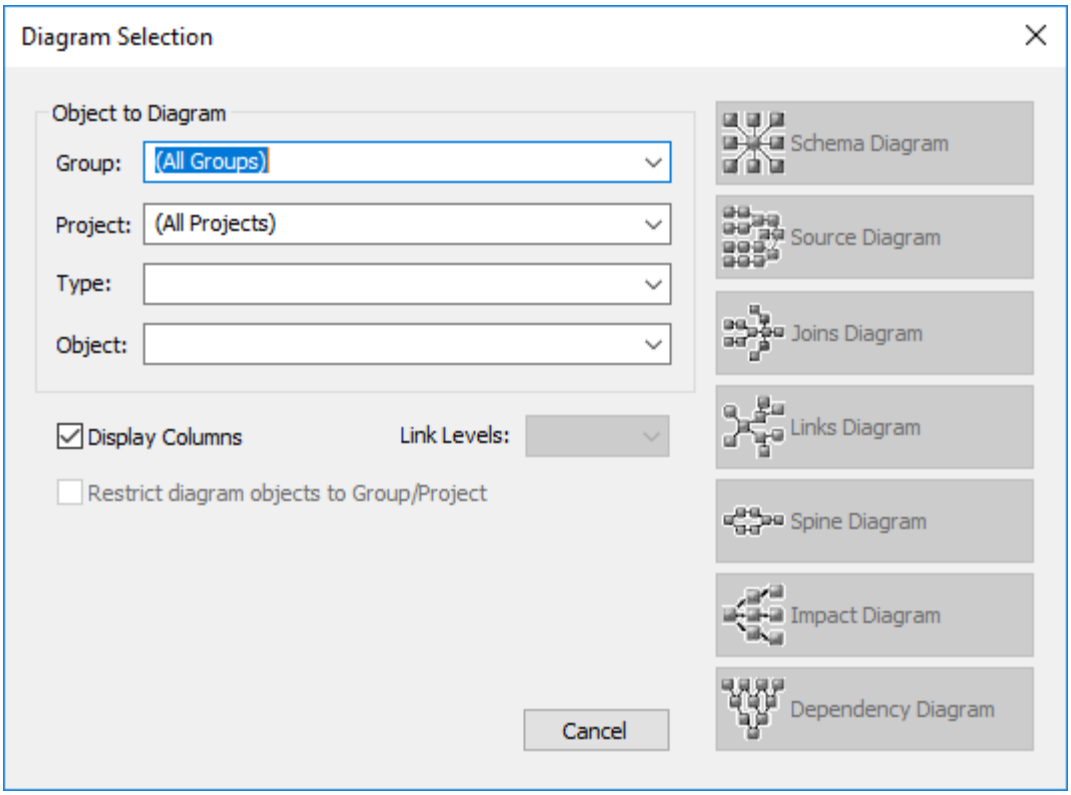
Diagrams

Types of Diagrams

Six types of diagrams are provided to give visual representation of what has been created. These are

- The **Schema Diagram**
- The **Source Diagram**
- The **Joins Diagram**
- The **Links Diagram**
- The **Impact Diagram**
- The **Dependency Diagram**

1. To display the **Diagram Selection** dialog, click on the diagrammatic view button

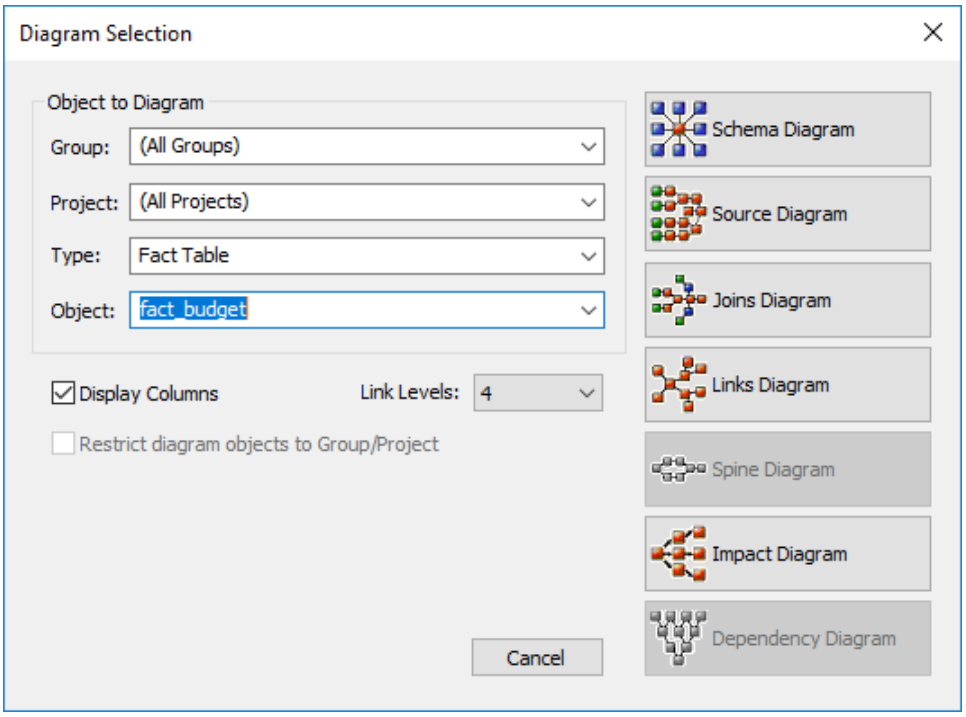


2. Choose the object to diagram by optionally choosing the **Type** to limit the selection list; and then selecting the **Object**. The diagram type buttons on the right will then become active and you can choose the type of diagram to display.

Note

Projects or groups that have been flagged as inactive are not displayed/available from the Group and Project drop-down lists. Refer to **Maintaining Group/Project Active Flag** for details.

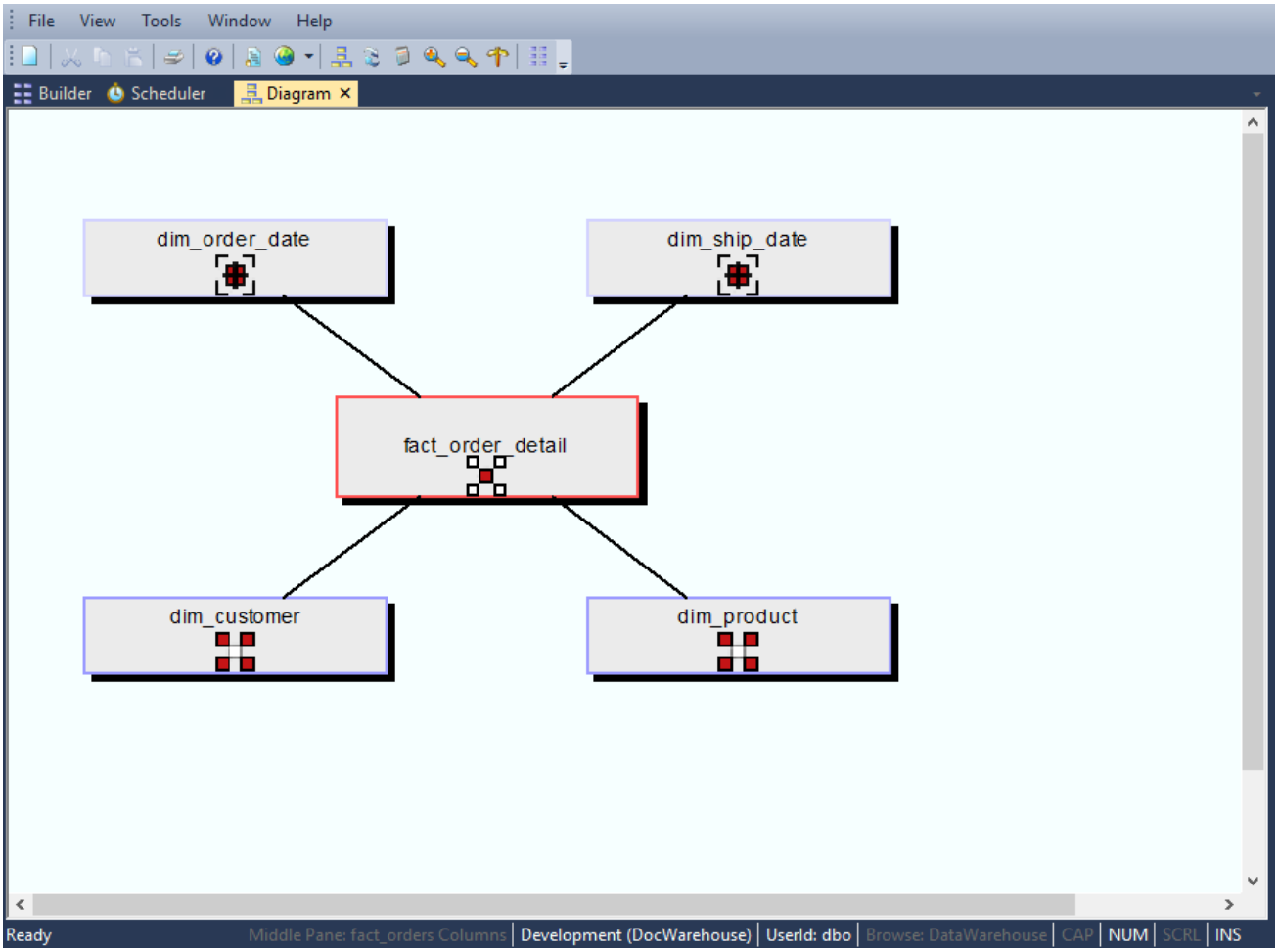
© Copyright, WhereScape Software Ltd All rights reserved.



Schema Diagram

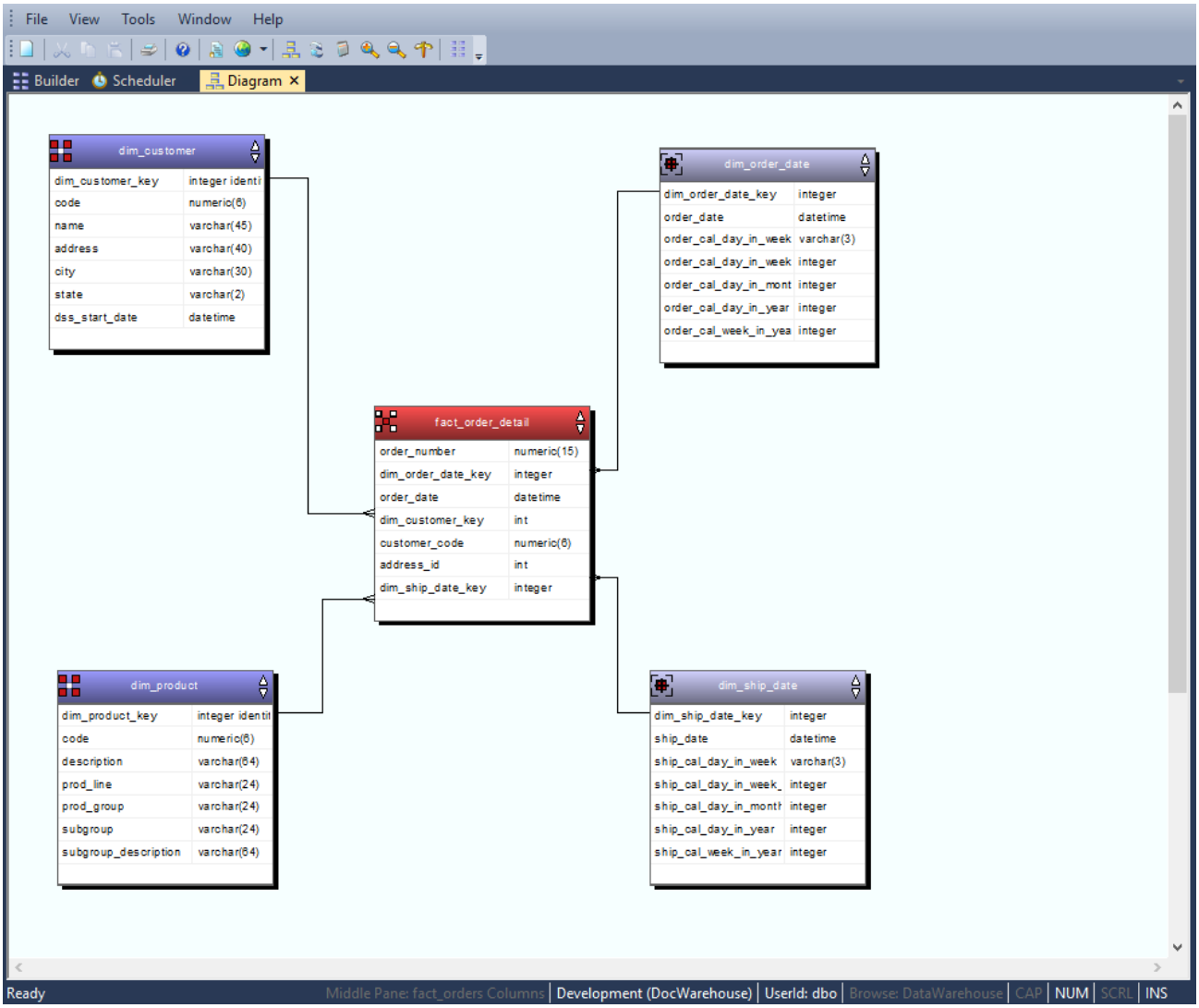
A star schema diagram can be displayed for a fact table, an aggregate table and an OLAP cube. It shows the central table with the outlying dimensions.

An example of a **Schema** diagram in **Standard Diagram** format is displayed below.



© Copyright, WhereScape Software Ltd All rights reserved.

An example of a **Schema** diagram in **Detail Diagram** format is displayed below.



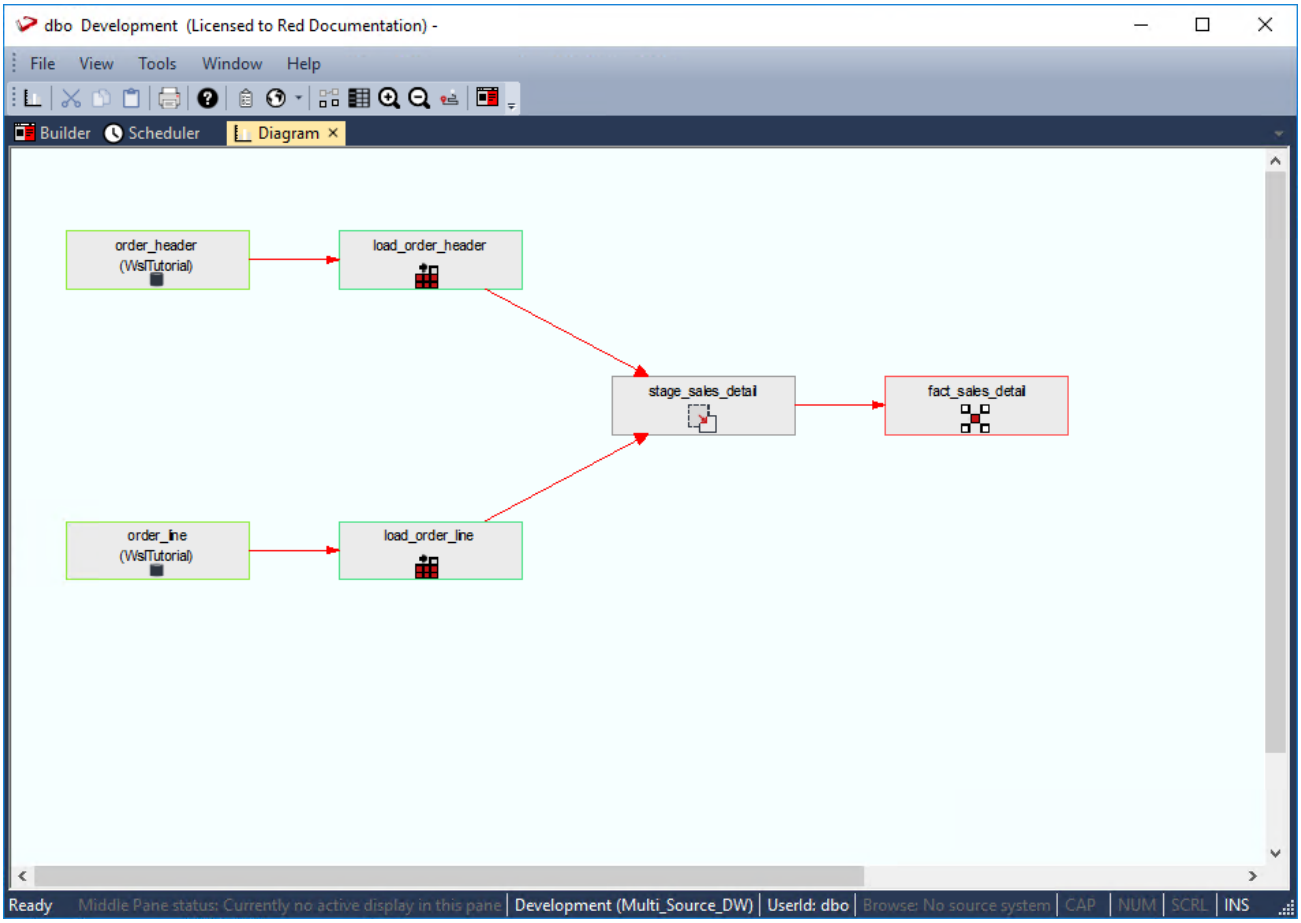
The star schema diagrams are produced in **Standard Diagram** format as part of the user and technical documentation when you select **Doc > Create Documentation** from the main builder window.

© Copyright, WhereScape Software Ltd All rights reserved.

Source Diagram

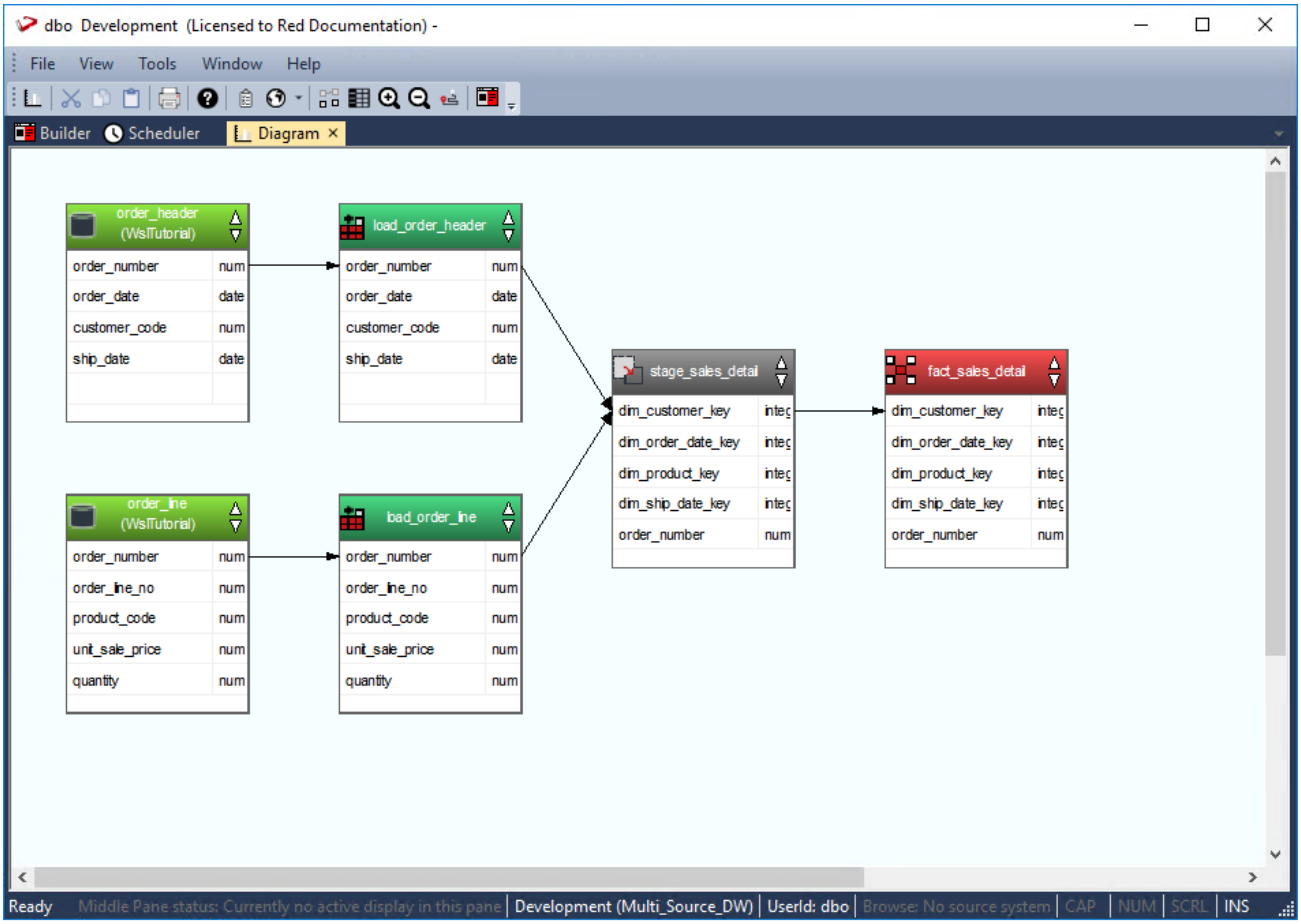
A source tracking diagram can be displayed for any table. It shows connections back from the chosen table to the source tables from which information was derived. Hovering the cursor over a line shows additional information. For lines going into load tables, the source of the data will be displayed; while for other lines in the diagram, the procedure used to move data between two tables is displayed.

An example of a **Source** diagram in **Standard Diagram** format is displayed below.



© Copyright, WhereScape Software Ltd All rights reserved.

An example of a **Source** diagram in **Detail Diagram** format is displayed below.

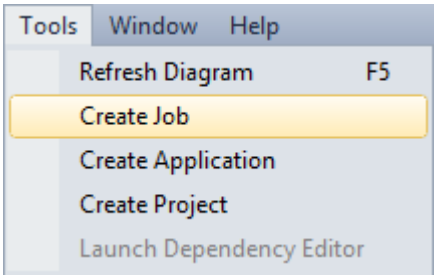


The Source diagrams are produced in **Standard Diagram** format as part of the technical documentation, when you select **Doc > Create Documentation** from the main Builder window.

Creating a Job from a Source Diagram

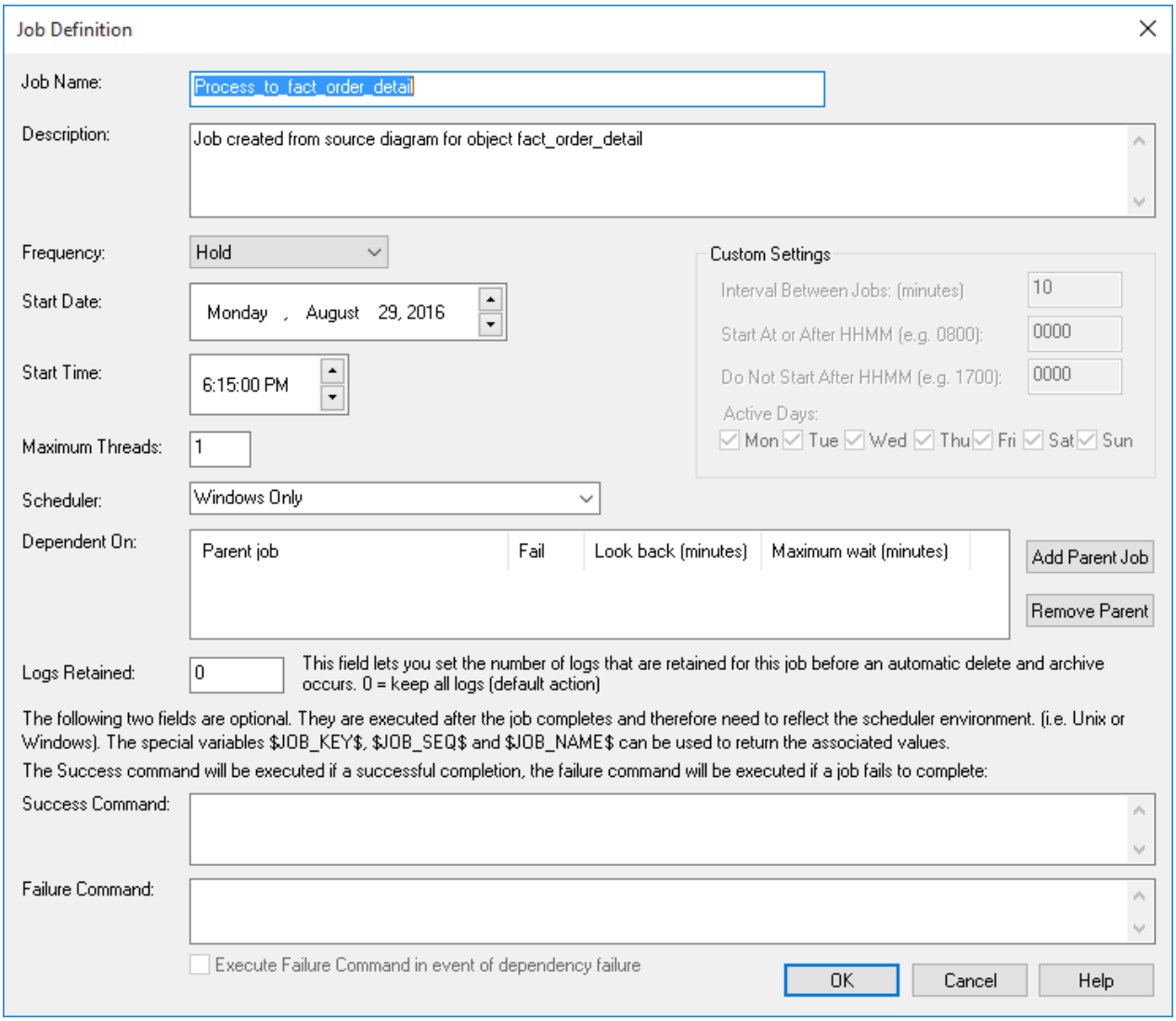
Once a source tracking diagram has been created for a table, a scheduler job can be generated from the diagram. This job will be called **Process_to_table_name**, where table_name is the name of the table the track back diagram was run for.

To create a Job, select **Create Job** from the **Tools** menu after the diagram is displayed:



© Copyright, WhereScape Software Ltd All rights reserved.

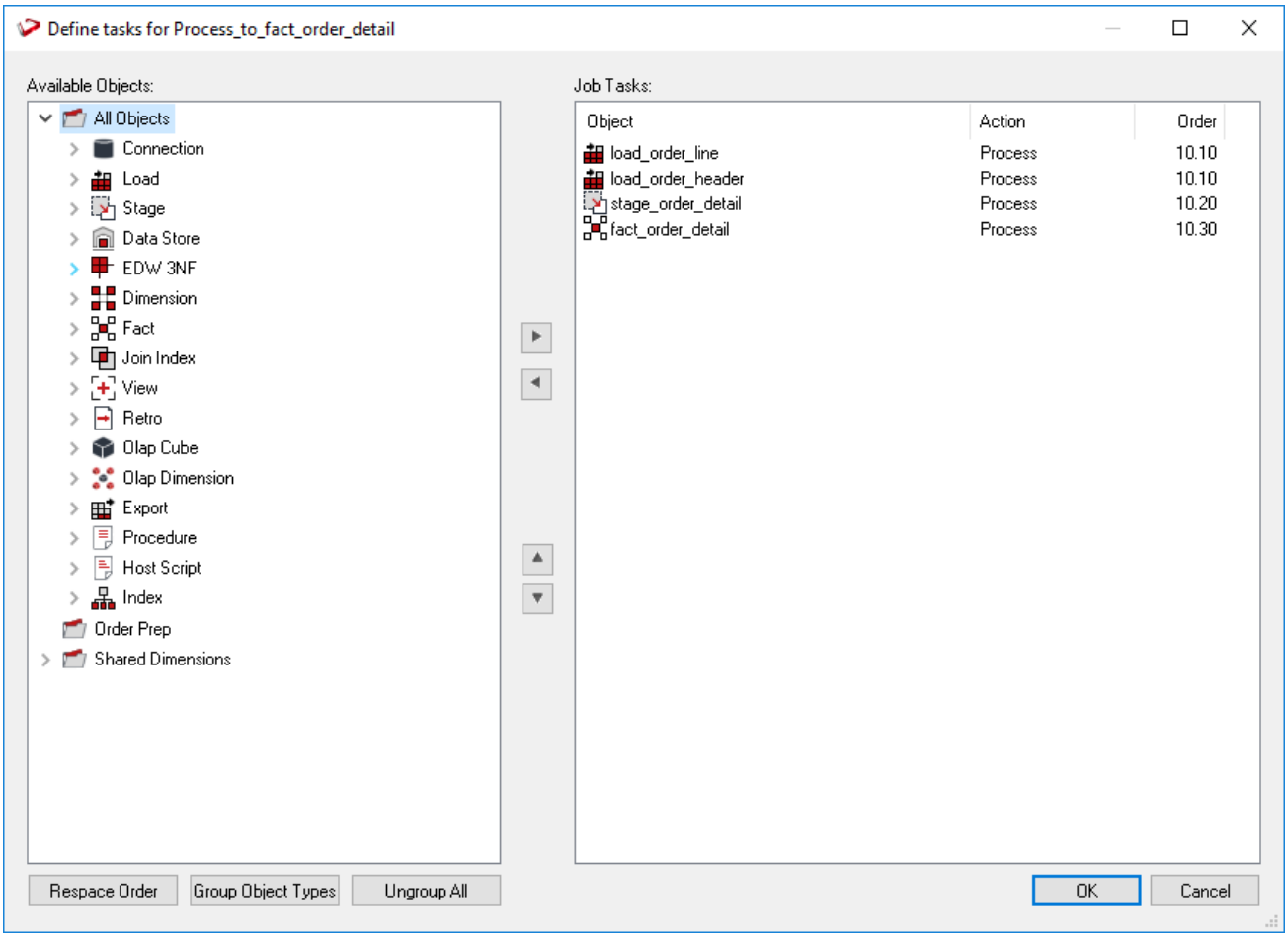
The Job Definition is then displayed:



Make any changes here that are required and click **OK**.

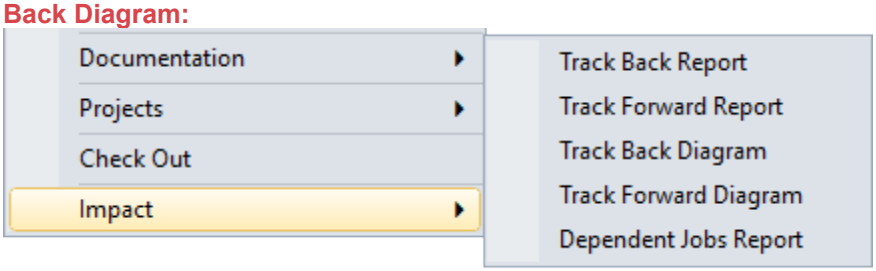
© Copyright, WhereScape Software Ltd All rights reserved.

For the diagram above, a job is created with the following **tasks**:



Note:

It is also possible to display the source diagram by right-clicking on a table and choosing **Impact>Track Back Diagram:**

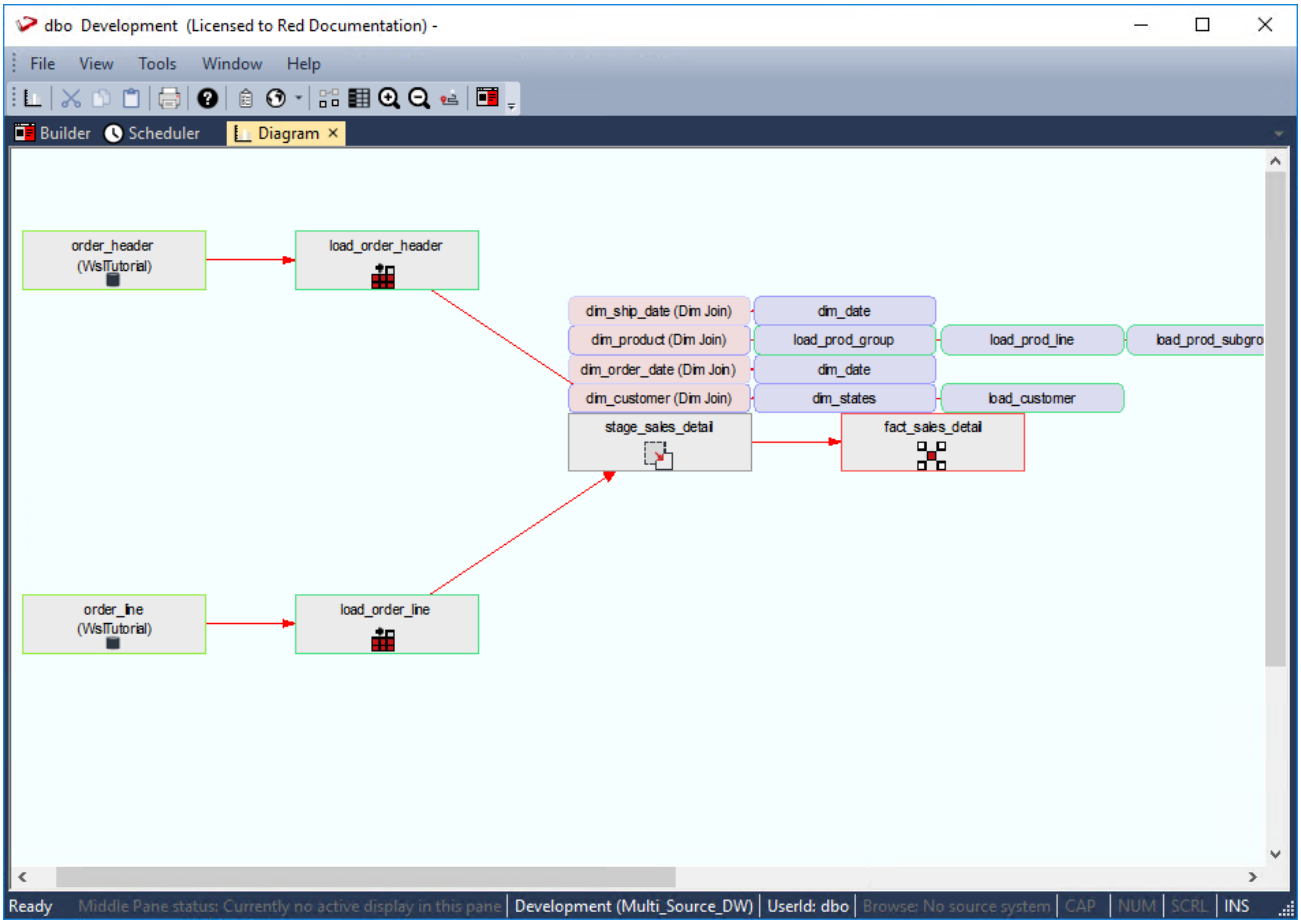


© Copyright, WhereScape Software Ltd All rights reserved.

Joins Diagram

A data join track back diagram can be displayed for any table. It shows connections back from the chosen table to the source tables from which information was derived and includes dimension table joins. Hovering the cursor over a line shows additional information. For lines going into load tables, the source of the data will be displayed; while for other lines in the diagram, the procedure used to move data between two tables is displayed.

An example of a **Joins** diagram in **Standard Diagram** format is displayed below.

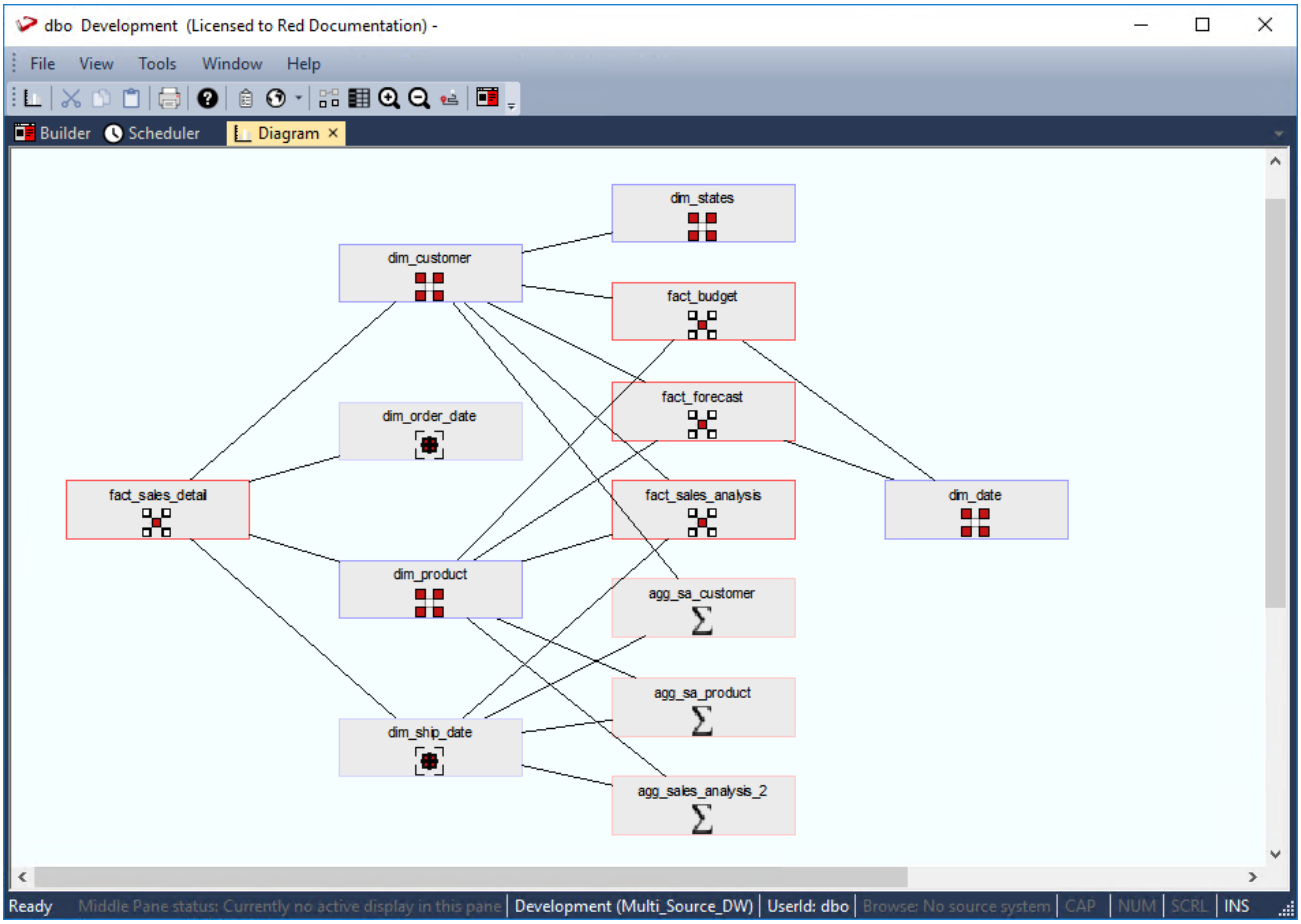


© Copyright, WhereScape Software Ltd All rights reserved.

Links Diagram

A linked tables diagram can be shown for any table. It shows relationships between tables, looking out from the chosen table a selected number of hops. The number of hops is determined by table relationships and source and target relationships.

An example of a **linked tables** diagram in **Standard Diagram** format is displayed below.



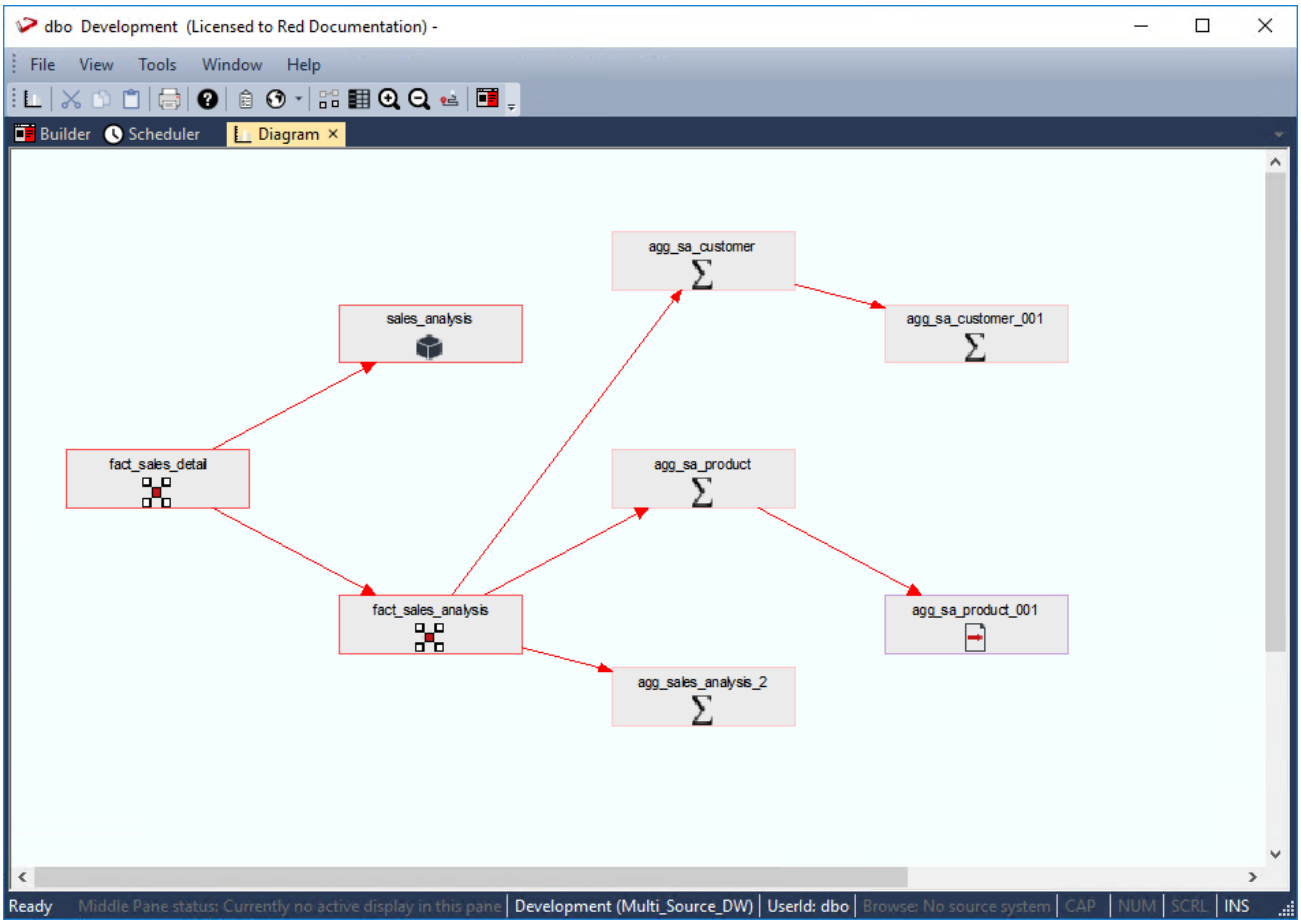
© Copyright, WhereScape Software Ltd All rights reserved.

Impact Diagram

A track forward impact diagram can be displayed for any table. It shows connections forward from the chosen table to the subsequent tables built with columns from this table.

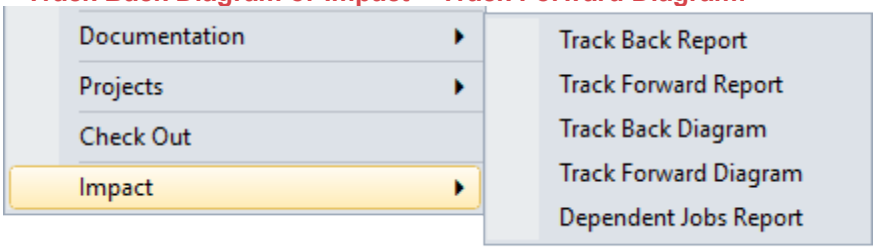
A track back impact diagram can be displayed for any table. It shows connections backwards from the chosen table to the previous tables.

An example of an **Impact** diagram in **Standard Diagram** format is displayed below.



Note

It is also possible to display the track back / forward diagram by right-clicking on a table and choosing **Impact > Track Back Diagram or Impact > Track Forward Diagram:**



© Copyright, WhereScape Software Ltd All rights reserved.

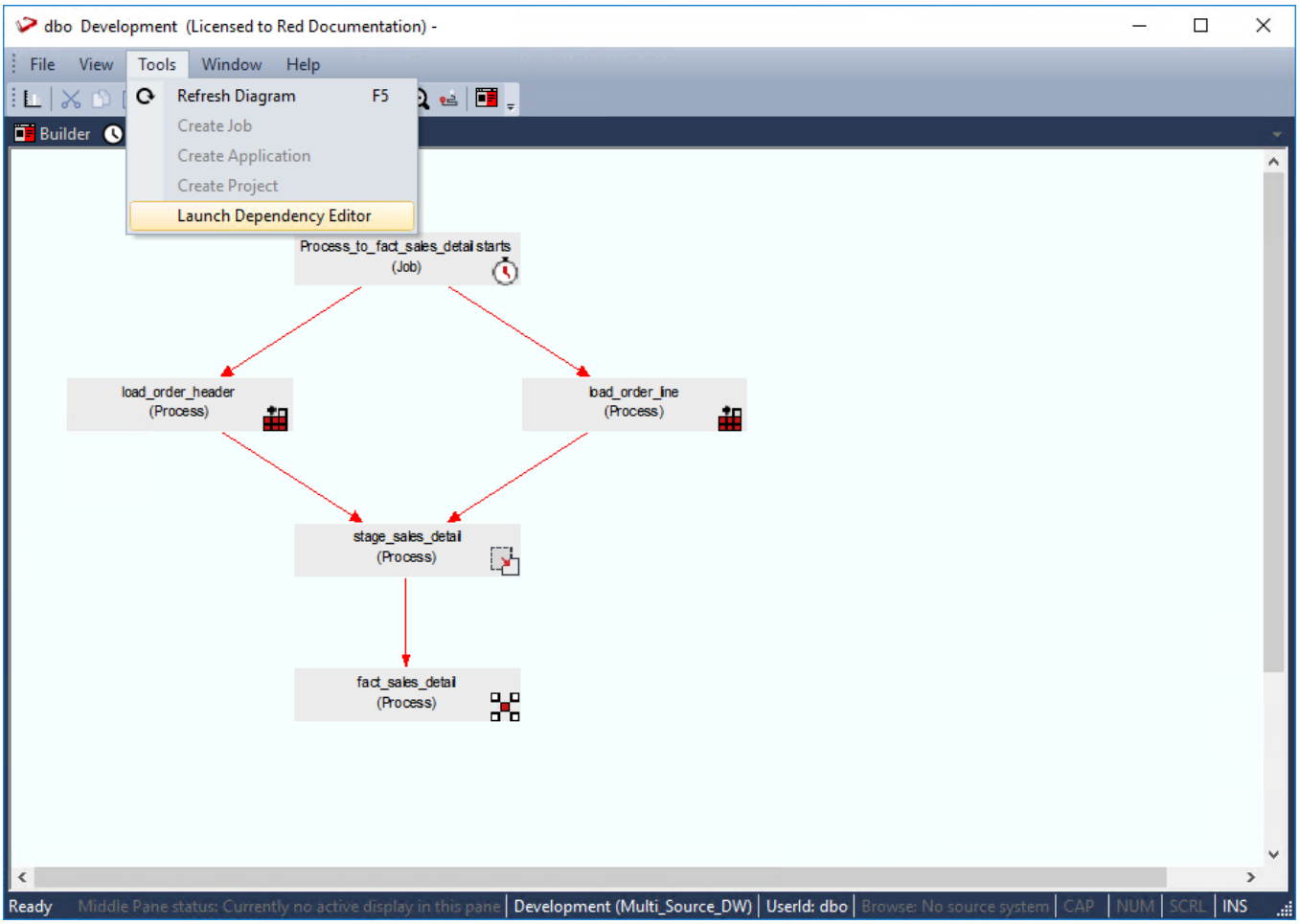
Dependency Diagram

A job dependency diagram can be displayed for any job defined in the WhereScape RED scheduler. It shows the parent and child relationships between tasks within a job.

An example of a **Dependency** diagram in **Standard Diagram** format is displayed below.

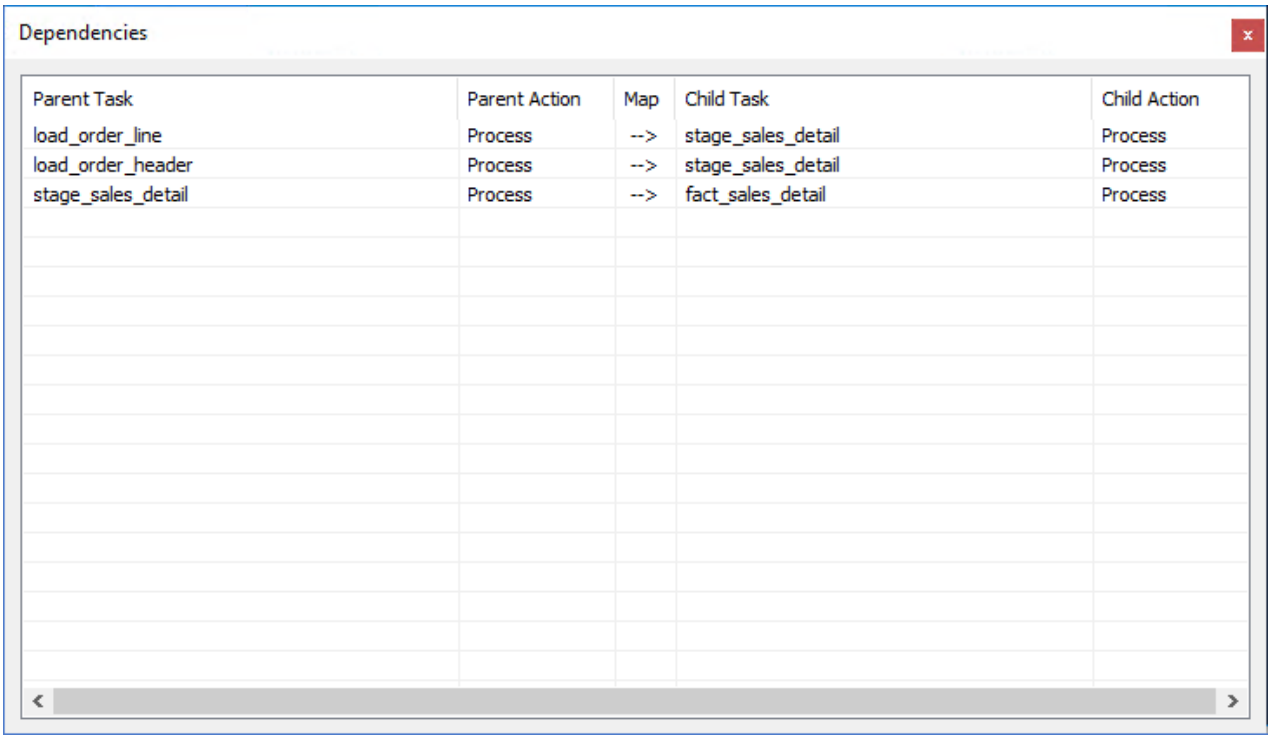
Editing a Job's Dependencies from a Job Dependency diagram

Once a Job Dependency diagram has been created for a job, its dependencies can be edited from the diagram. To do this, select **Launch Dependency Editor** from the **Tools** menu:



© Copyright, WhereScape Software Ltd All rights reserved.

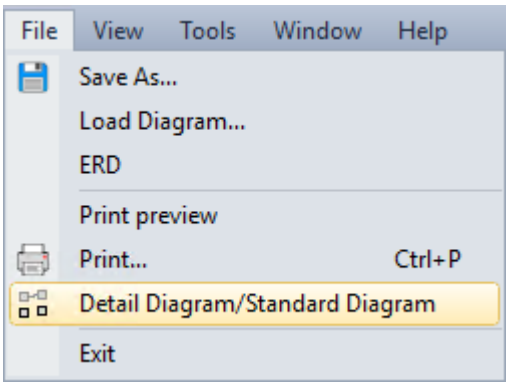
The **Dependencies** window enables you to edit the dependencies within a job:



Working with Diagrams

Diagram Display

Once displayed there are two modes for diagram display; standard and detailed. To move between displays, select **File > Detail Diagram / Standard Diagram**



or use the **toggle** button.

Diagram Save

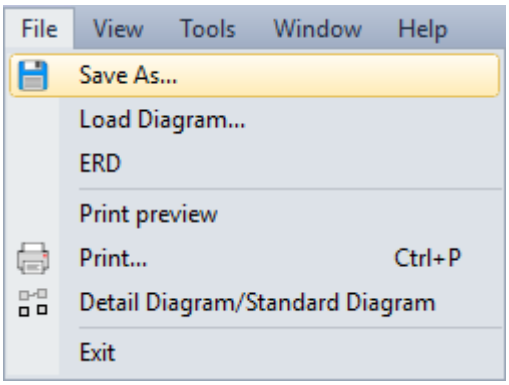
A diagram can be saved either as a meta file or as a jpeg image. If saved as a meta file it can be subsequently reloaded and re-edited.

Note

A jpeg cannot be reloaded into WhereScape RED.

© Copyright, WhereScape Software Ltd All rights reserved.

The diagram can be saved by selecting **File > Save As ...**



Note

By default, the diagram is saved as a meta file.

Diagram Load

If a diagram has been saved as a meta file, it can be reloaded. To reload a saved meta file, switch to diagrammatic view and then select the menu option **File > Load Diagram**. A dialog box will allow you to choose a windows meta file (*.wmf). If the meta file had previously been saved from WhereScape RED then the diagram will be loaded.

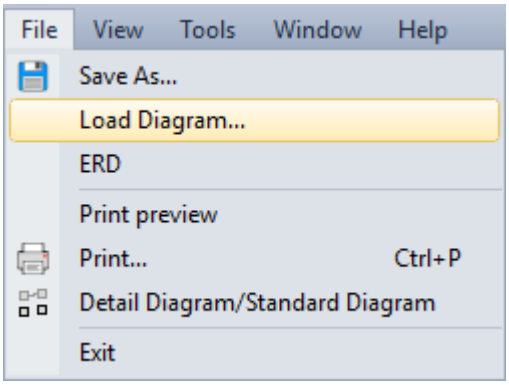
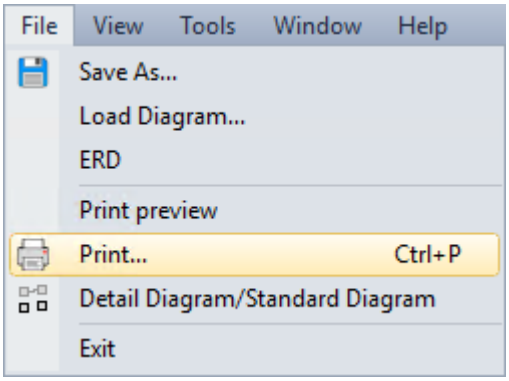


Diagram Print

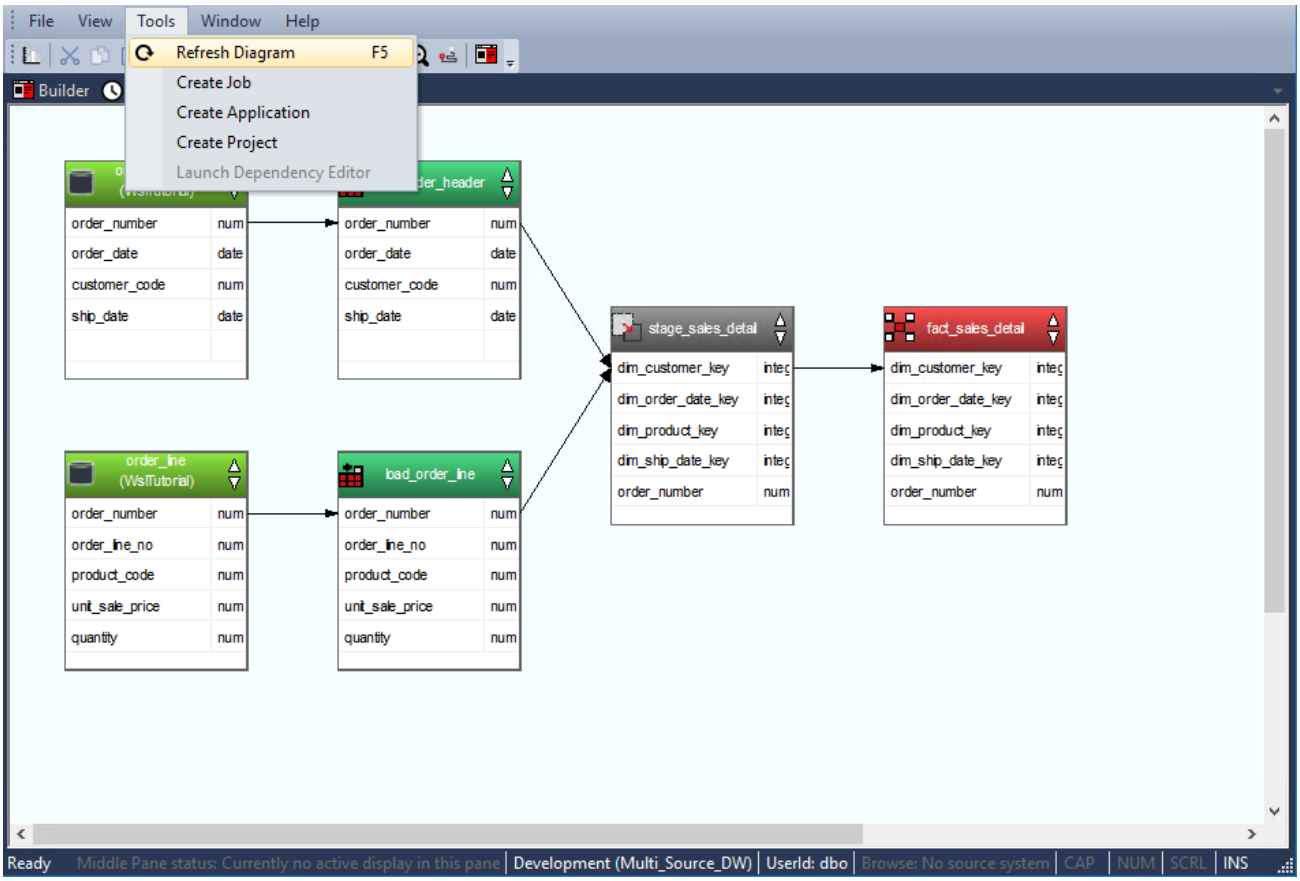
A diagram can be printed by selecting **File > Print ...**



© Copyright, WhereScape Software Ltd All rights reserved.

Diagram Refresh

Once a diagram has been displayed, it can be refreshed by choosing **Tools > Refresh Diagram**, or by pressing **F5**.

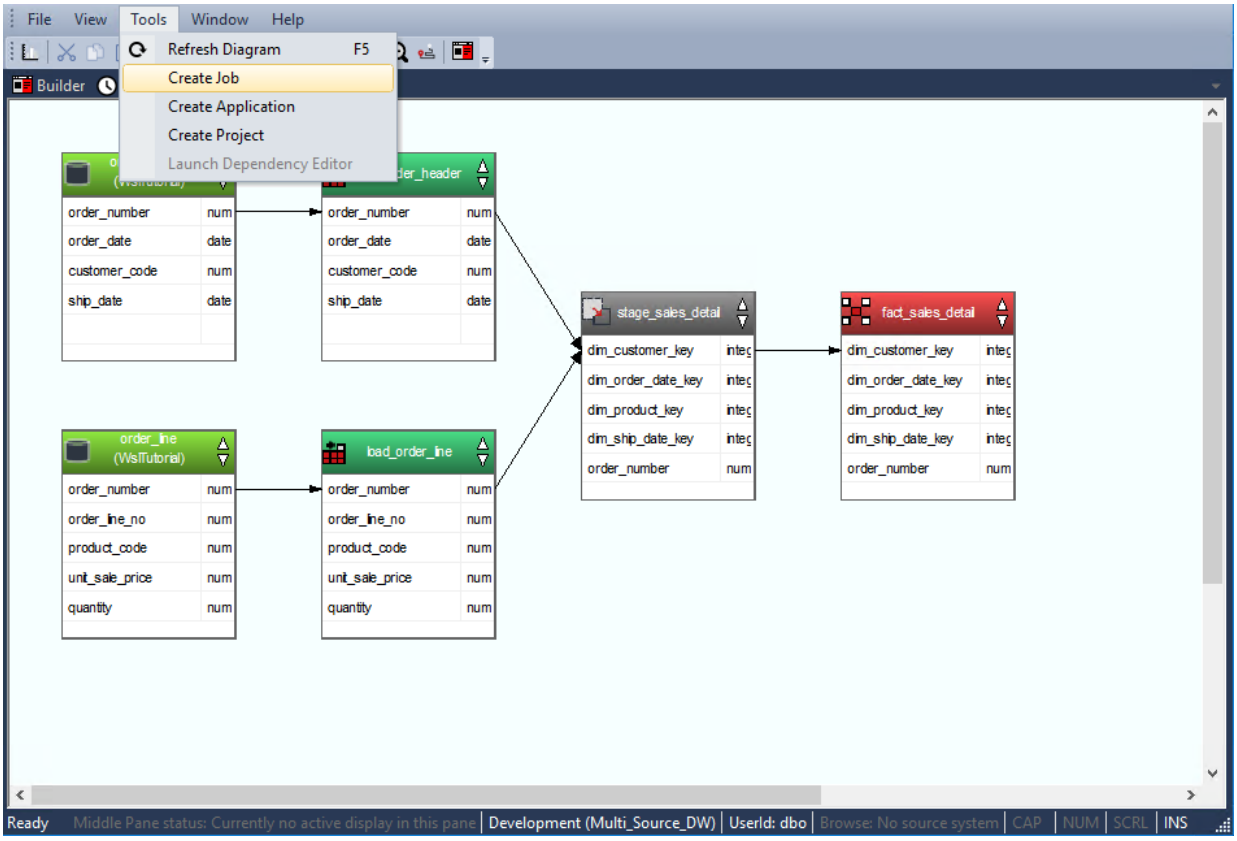


© Copyright, WhereScape Software Ltd All rights reserved.

Creating a Job from a Diagram

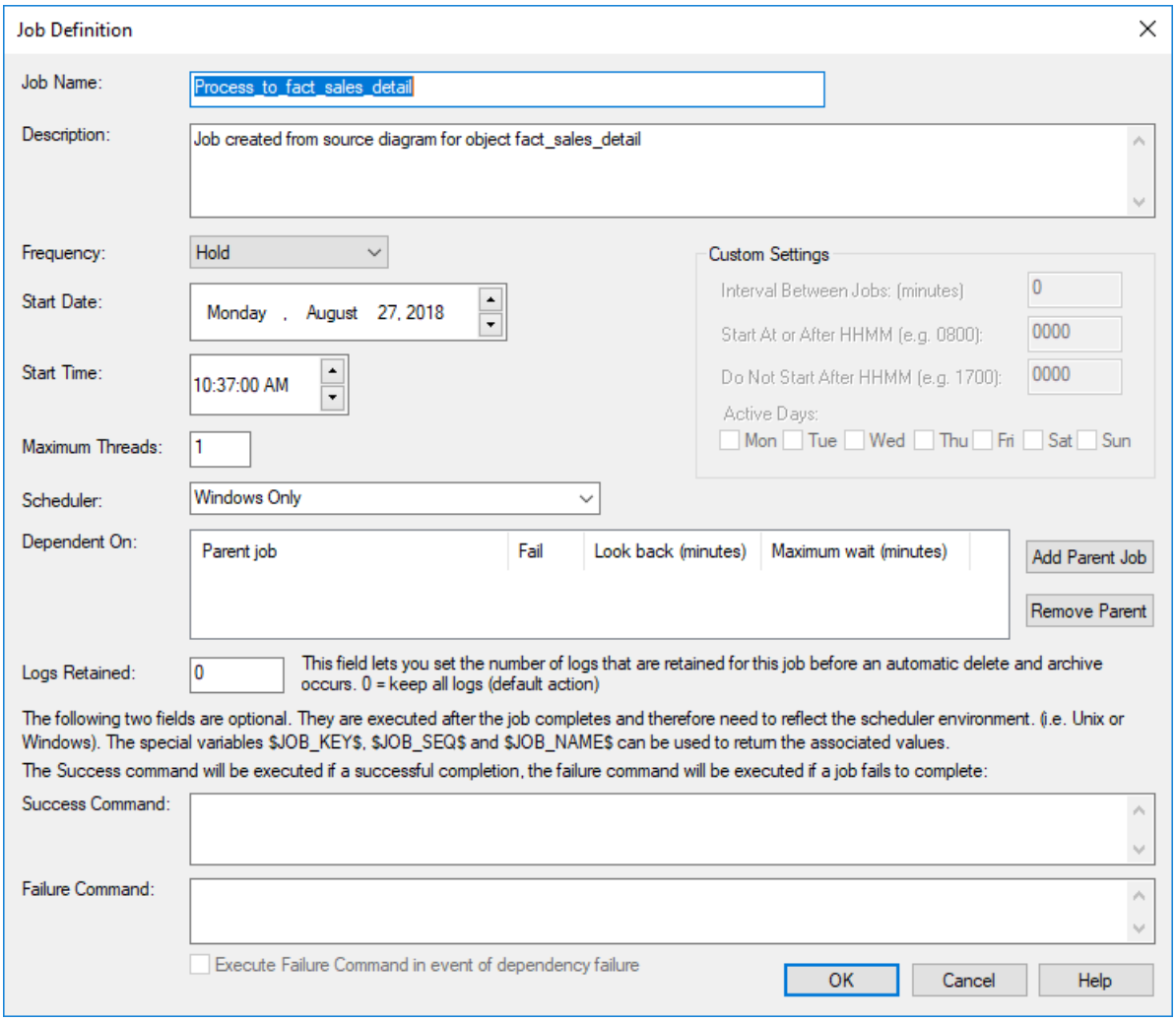
A job can be created from a **Source** diagram or a **Joins** diagram.

1. Once the diagram is displayed, select **Tools > Create Job**.



2. Edit the job as required and then click **OK**.

© Copyright, WhereScape Software Ltd All rights reserved.

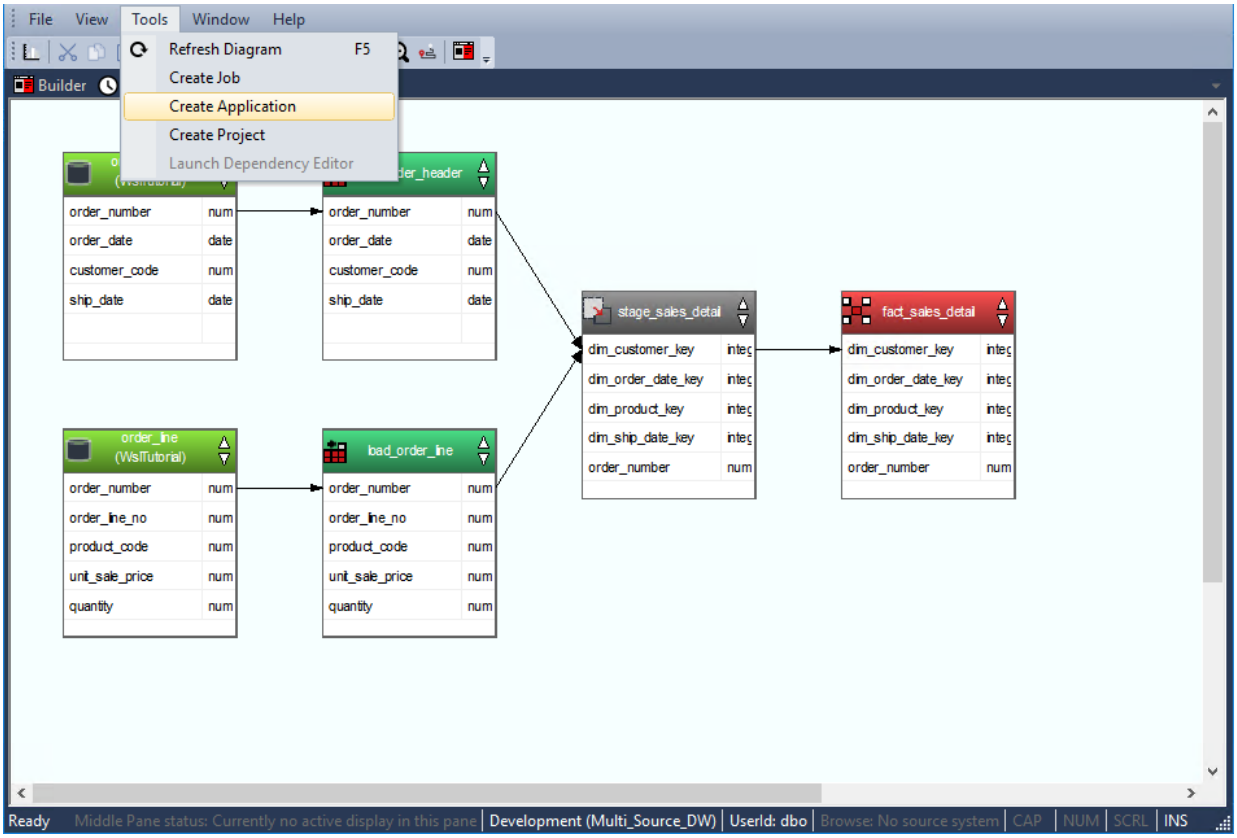


© Copyright, WhereScape Software Ltd All rights reserved.

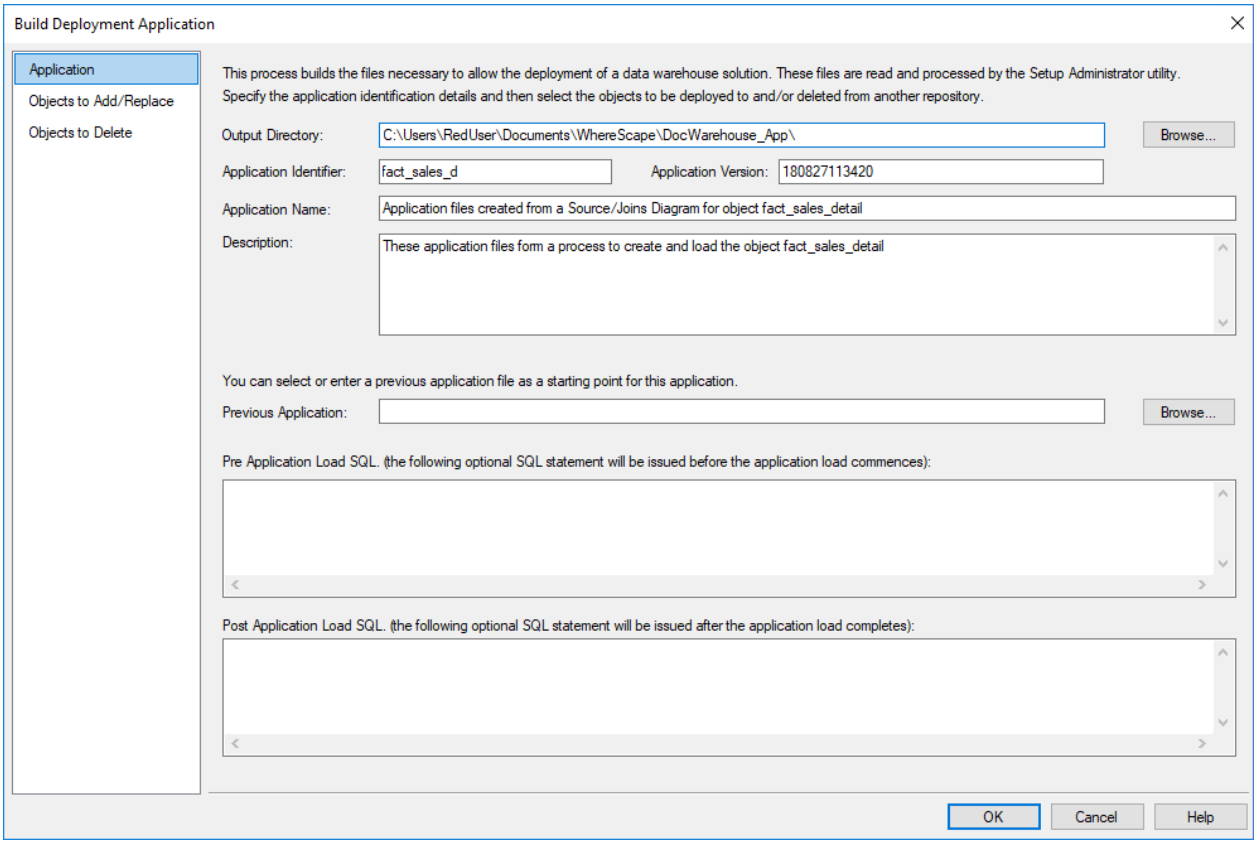
Creating an Application from a Diagram

An application can be created from a **Source** diagram or a **Joins** diagram.

1. Once the diagram is displayed, select **Tools > Create Application**.

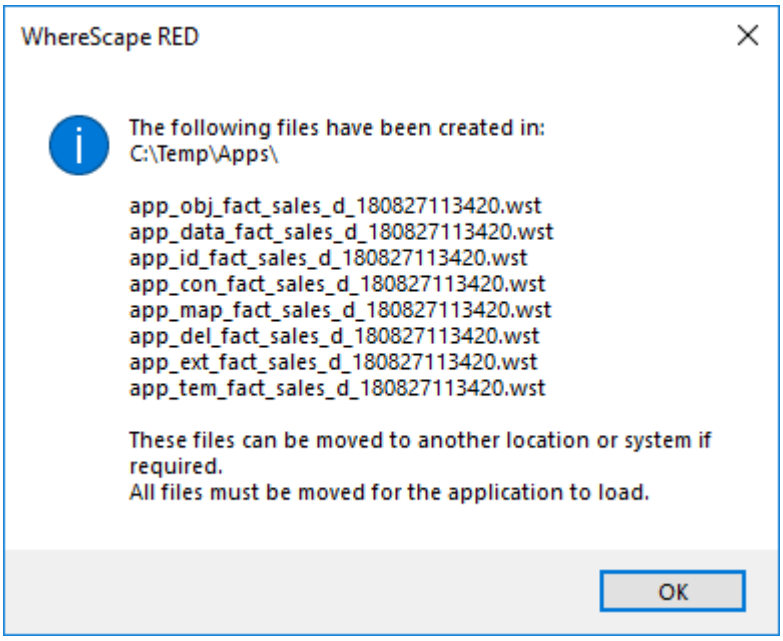


2. Edit the application as required and then click **OK**.



© Copyright, WhereScape Software Ltd All rights reserved.

3. A message is displayed, confirming the creation of the application files. Click **OK**.



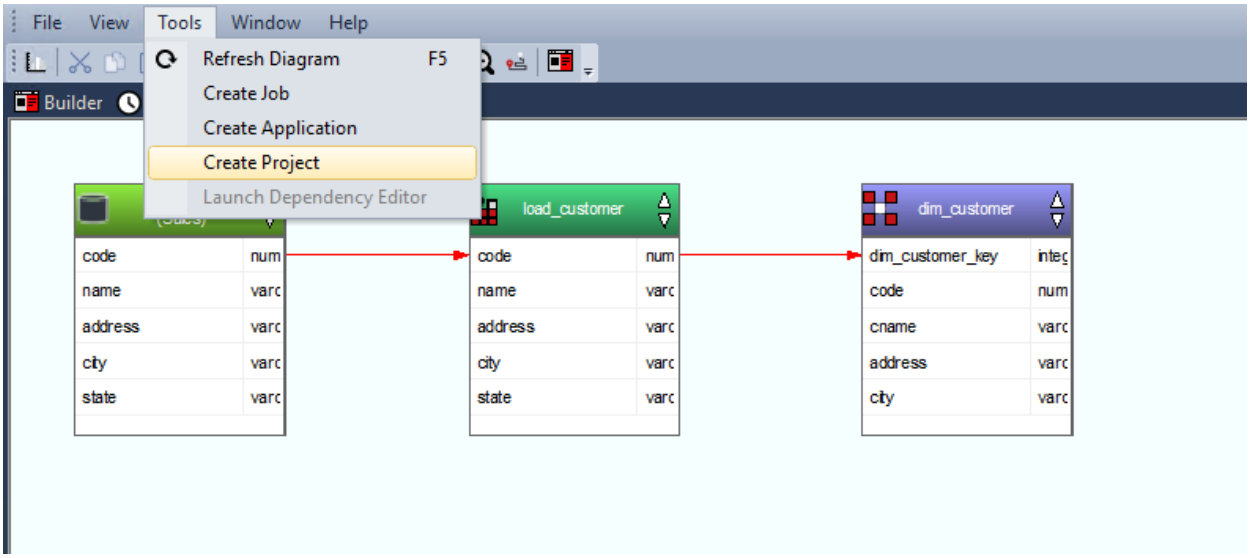
Note

Creating an application from a diagram saves the objects in the diagram and the associated objects, including indexes.

Creating a Project from a Diagram

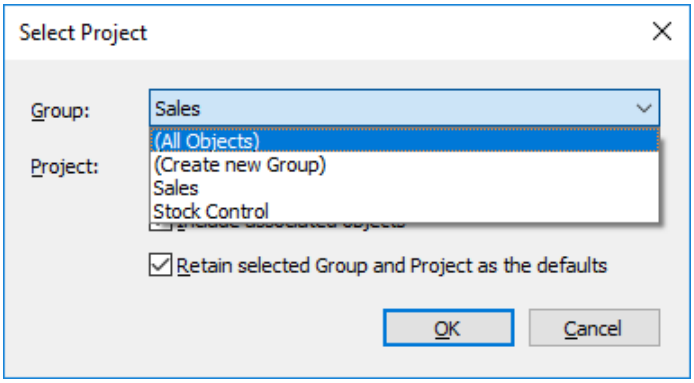
A project can be created from a **Source** diagram or a **Joins** diagram.

1. Once the diagram is displayed, select **Tools > Create Project**.

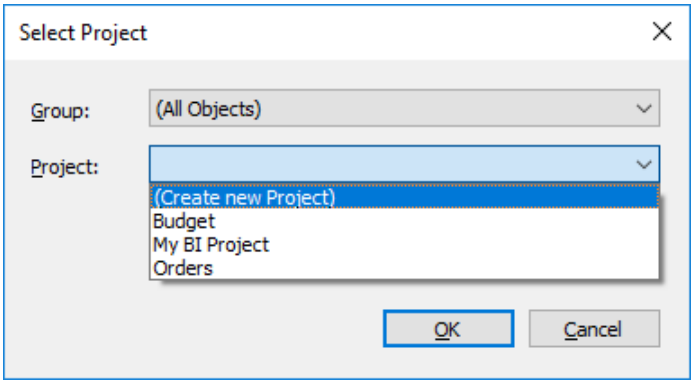


© Copyright, WhereScape Software Ltd All rights reserved.

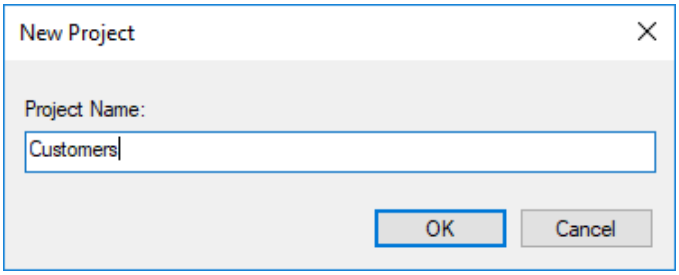
2. Select an existing project group or create a new group.



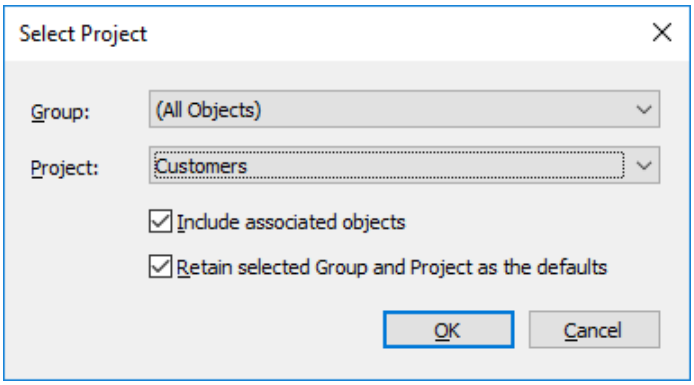
3. Select **Create new Project**.



4. Type in the name of the new project and then click **OK**.



5. Click **OK** in the **Select Project** window.



The objects in the diagram are moved into the selected project.

If the **Include Associated Objects** check box is selected, all associated procedures, scripts and indexes are included. This option is selected by default.

If the **Retain selected Group and Project as the defaults** check box is selected, the selected Group and Project are retained as the defaults. This option is selected by default.

© Copyright, WhereScape Software Ltd All rights reserved.

Query Tool Access

Where the business definition has been set in the properties of either a fact or dimension column, WhereScape RED will attempt to record this information in the database. For the Oracle database, this business definition is stored as a column comment. Many end user query tools can make use of these Oracle column comments. They can also be examined via the Oracle database system views all_col_comments and user_col_comments. For the SQL Server database, the business definition is also stored as a column comment via the sp_addextendedproperty procedure with a property name of **Comment**.

In addition, a number of metadata views exist to make the task of connecting an end user query tool to the data warehouse simpler. Many end user tools can make use of external data to populate their end-user-layer (eul) definitions. Using these views, end user query tools can utilize WhereScape RED's business metadata. The views supplied are:

Dimension tables and columns

The view **ws_admin_v_dim_col** provides a listing of all columns in all dimensions that have the 'Eul visible' flag set. Included in the information provided is the business definition and description of the column as defined within WhereScape RED.

Fact tables and columns

The view **ws_admin_v_fact_col** provides a listing of all columns in all fact tables that have the 'Eul visible' flag set. Included in the information provided is the business definition and description of the column as defined within WhereScape RED.

Fact - Dimension Joins

The view **ws_admin_v_fact_join** provides a listing of all fact tables and all the dimensions that each fact table joins to. Both the fact table dimension key column and the dimension key column are included in the view.

© Copyright, WhereScape Software Ltd All rights reserved.

Reports

WhereScape RED includes reports for analyzing columns, tables, procedures, indexes, objects and jobs. When these reports are run, the results are displayed in a separate tab in the bottom pane of the RED screen. The following sections describe the purpose, parameters and results for each report.

Dimension-Fact Matrix

This report shows dimension tables used by each fact and aggregate table in the metadata as a matrix.

Objects Included

The following WhereScape RED object types are included in this report:

- Dimension Tables
- Fact Tables
- Aggregate Tables

Parameters

There are no parameters for this report.

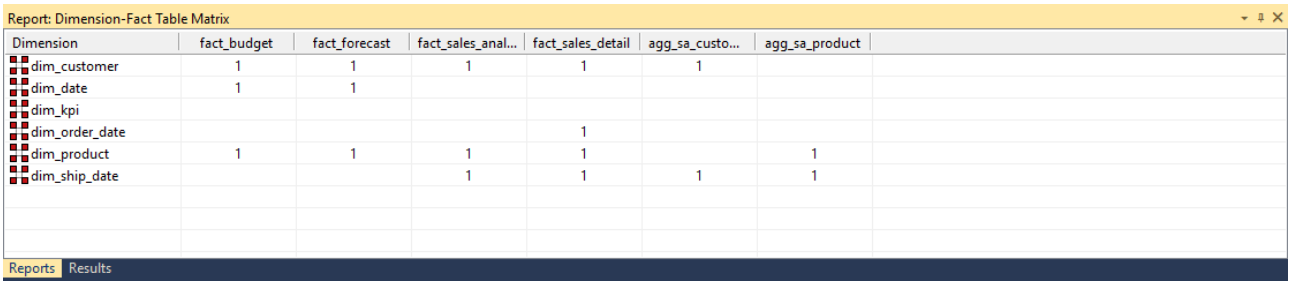
Results

The results of this report are displayed as a list of refreshed objects in the metadata repository with the following columns:

- Dimensions *(the dimension name)*
- Fact/Aggregate Table 1
- Fact/Aggregate Table 2
- Fact/Aggregate Table n

The cells in the cross tab have a 1 to indicate the dimension is used by the fact or aggregate table and blank otherwise. The result set is not sortable.

Report Example



Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

OLAP Dimension-Cube Matrix

This report shows the relationships between cube measure groups and OLAP dimensions in the metadata as a matrix.

Objects Included

The following WhereScape RED object types are included in this report:

- OLAP Dimensions
- OLAP Measure Groups

Parameters

There are no parameters for this report.

© Copyright, WhereScape Software Ltd All rights reserved.

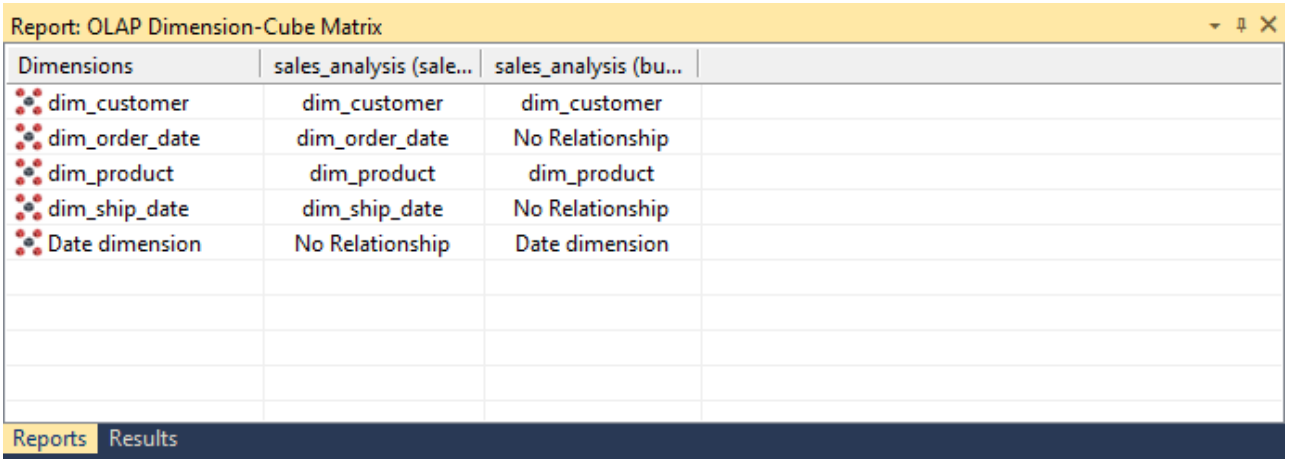
Results

The results of this report are displayed as a list of refreshed objects in the metadata repository with the following columns:

- Dimensions *(the dimension name)*
- Measure Group 1
- Measure Group 2
- ...
- Measure Group n

The cells in the cross tab have a value to indicate the relationship else 'No Relationship' if no relationship exists between the Measure Group and the OLAP Dimension. The result set is not sortable.

Report Example



Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Dimension Views of a Specified Dimension

This report shows dimension views built on a specified Dimension table.

Objects Included

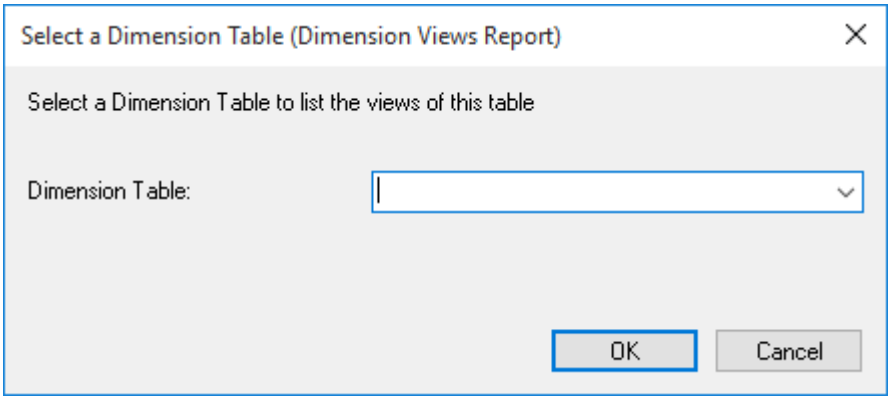
The following WhereScape RED object types are included in this report:

- Dimensions (Dimension View table type only)

Parameters

This report has one parameter:

- Dimension table name



© Copyright, WhereScape Software Ltd All rights reserved.

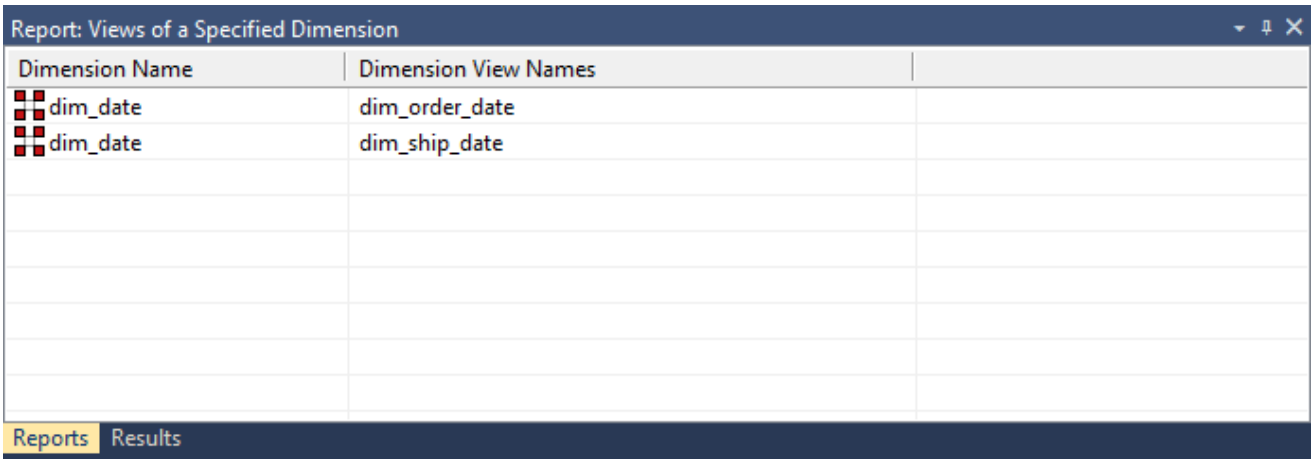
Results

The results of this report are displayed as a list of objects in the metadata repository with the following columns:

- Dimension Name *(the name of the dimension table)*
- Dimension View Names

The result set is sortable by clicking on the appropriate column heading.

Report Example



Sending Results to Microsoft Excel

Right-click on the result set and click **Output to File** to send the results of this report to Microsoft Excel.

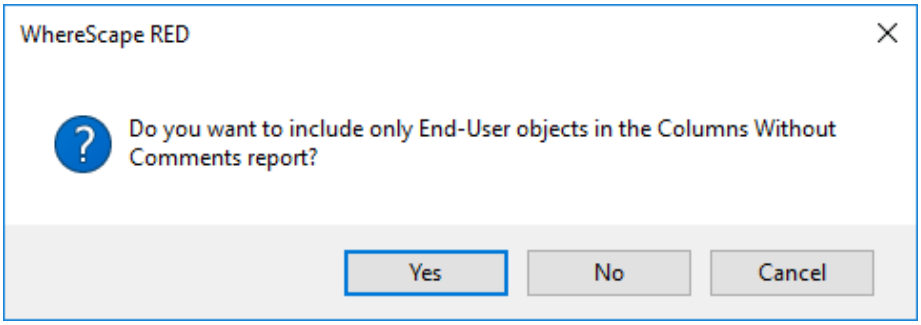
Column Reports

There are five reports for analyzing Columns:

- Columns Without Comments
- All Column Transformations
- Re-Usable Column Transformations
- Column Track-Back
- Column Track-Forward

Columns without Comments

This report shows **table** object columns in the metadata that don't have descriptions. The report prompts the user with the option to only display user facing table object columns or all table object columns.



Objects Included

The following WhereScape RED object types are included in this report:

- Dimension Tables and Views
- Fact Tables
- Aggregate Tables
- Hubs, Satellites and Links

© Copyright, WhereScape Software Ltd All rights reserved.

Note

The EUL object are set in **Tools > Options > Object Types > Object Type End User Setting.**

Parameters

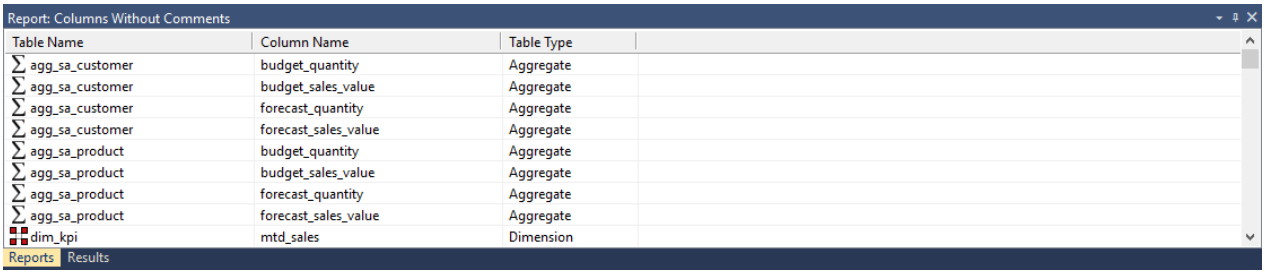
There are no parameters for this report.

Results

The results of this report are displayed as a list of objects and columns in the metadata that are missing comments with the following columns:

- Table name *(the name of the table)*
- Column Name
- Table type

Report Example



Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

All Column Transformations

This report shows all columns that have a column transformation and the details of the transformation.

Objects Included

The following WhereScape RED object types are included in this report:

- Load Tables
- Stage Tables
- Dimension Tables and Views
- Fact Tables
- Aggregate Tables
- Exports
- Hubs, Satellites and Links

Parameters

There are no parameters for this report.

Results

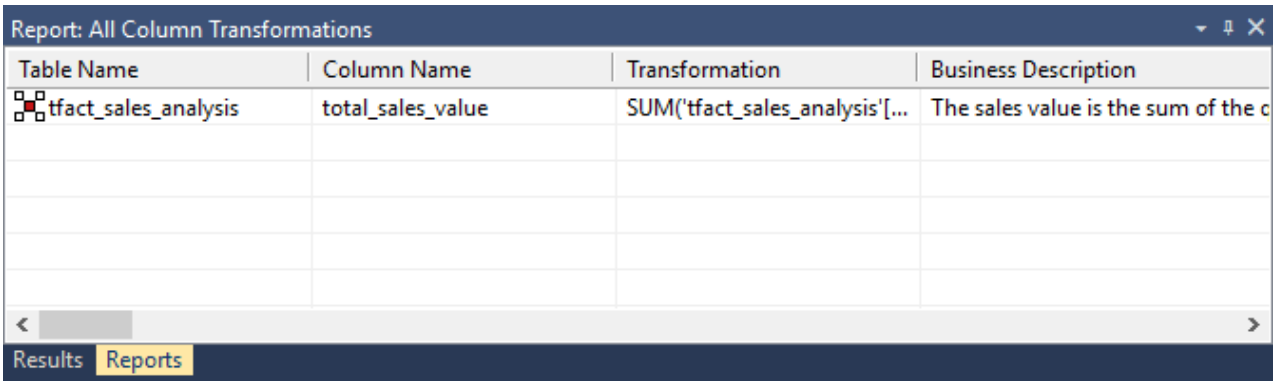
The results of this report are displayed as a list of objects in the metadata repository with the following columns:

- Table name *(the name of the table)*
- Column name
- Transformation

The result set is sortable by clicking on the appropriate column heading.

© Copyright, WhereScape Software Ltd All rights reserved.

Report Example



Sending Results to Microsoft Excel

Right-click on the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Re-Usable Column Transformations

This report shows all reusable transformations as defined via **Tools > Define Re-usable Transformations**.

Parameters

There are no parameters for this report.

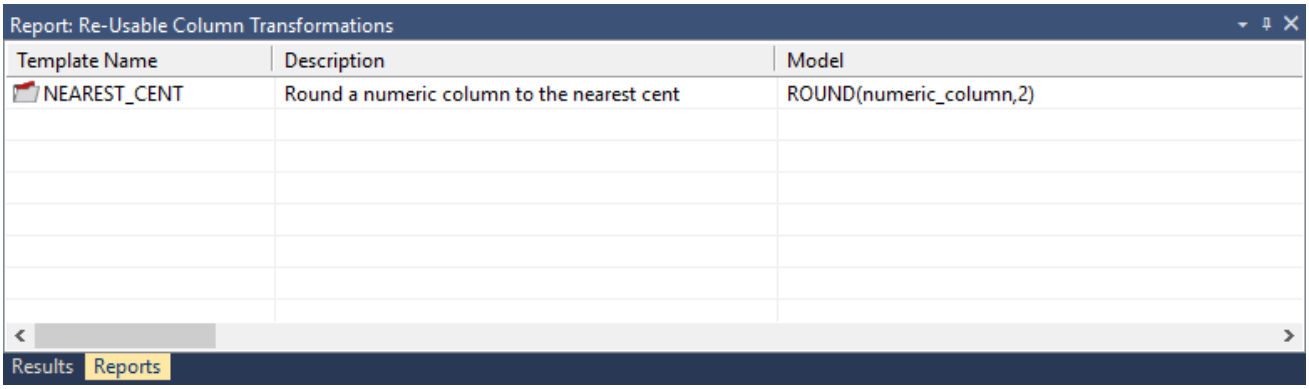
Results

The results of this report are displayed as a list of Re-Usable transformations with the following columns :

- Template Name
- Description

The result set is not sortable.

Report Example



Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Column Track-Back

This report lists the origins of the selected objects.

Objects Included

The following WhereScape RED object types are included in this report:

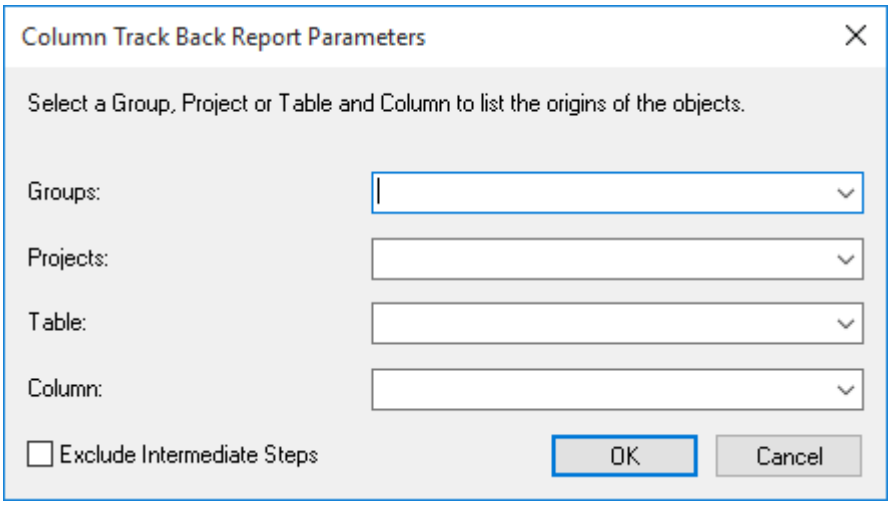
- Load Tables
- Stage Tables
- Dimension Tables and Views
- Fact Tables

© Copyright, WhereScape Software Ltd All rights reserved.

- Aggregate Tables
- Exports
- Hubs, Satellites and Links

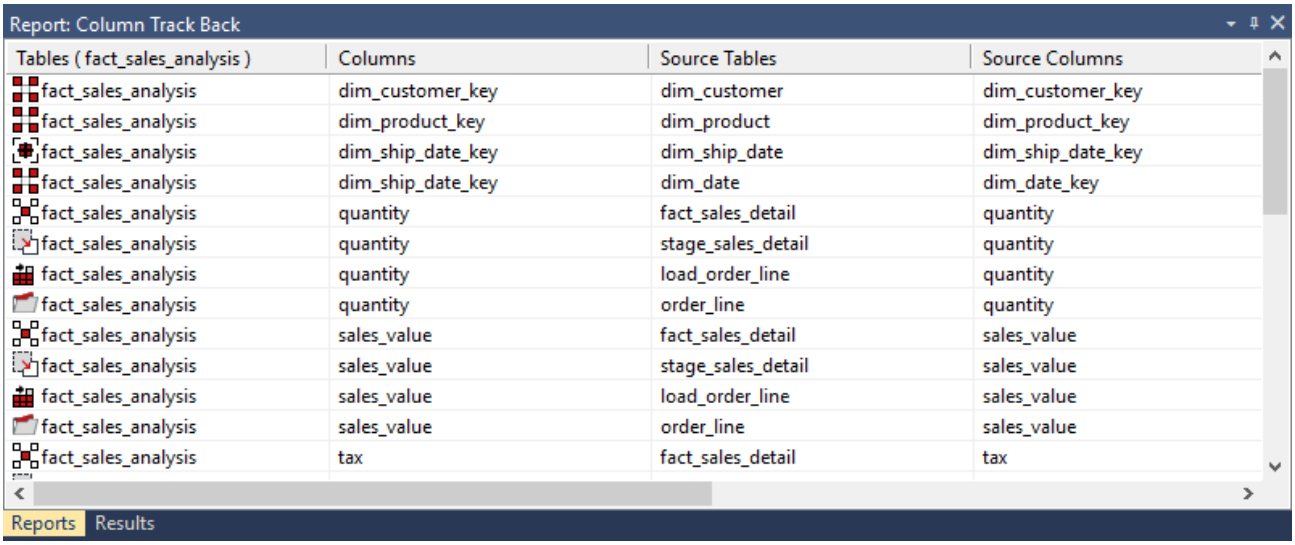
Parameters

- Groups
- Projects
- Table
- Column



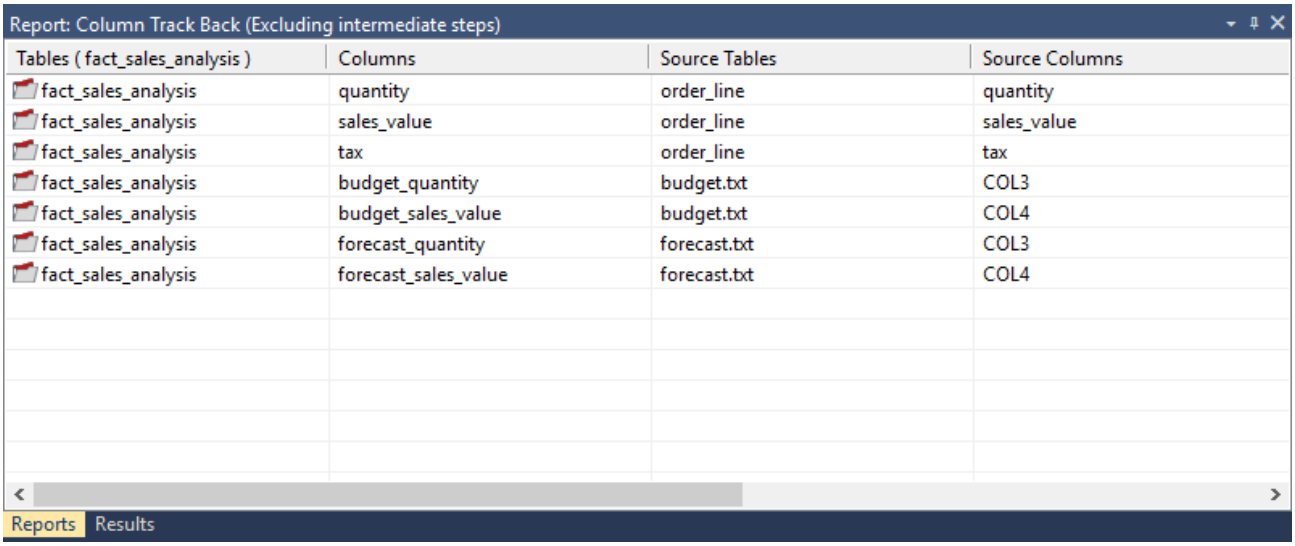
Results

If you left the **Exclude Intermediate Steps** check box **unchecked**, then the results screen is as follows, showing the line of origins for the selected tables:



© Copyright, WhereScape Software Ltd All rights reserved.

If however, you selected **Exclude Intermediate Steps**, then the results screen is as follows, showing only the original source table for the selected tables:



The results of this report are displayed as a list of source tables and columns, the order of the result set determining the immediate lineage. The report includes the following columns:

- Tables *(the name of a selected table)*
- Columns *(the name of a selected column)*
- Source Tables
- Source Columns

The result set is **not** sortable, as the order of the result set determines the immediate lineage.

Sending Results to Microsoft Excel

Right-click on the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Column Track-Forward

This report lists the columns derived from the selected objects.

Objects Included

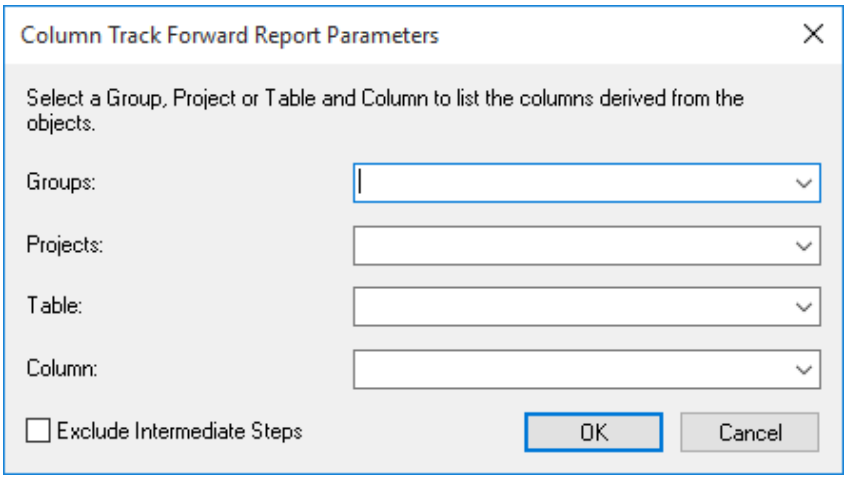
The following WhereScape RED object types are included in this report:

- Load Tables
- Stage Tables
- Dimension Tables and Views
- Fact Tables
- Aggregate Tables
- Exports
- Hubs, Satellites and Links

Parameters

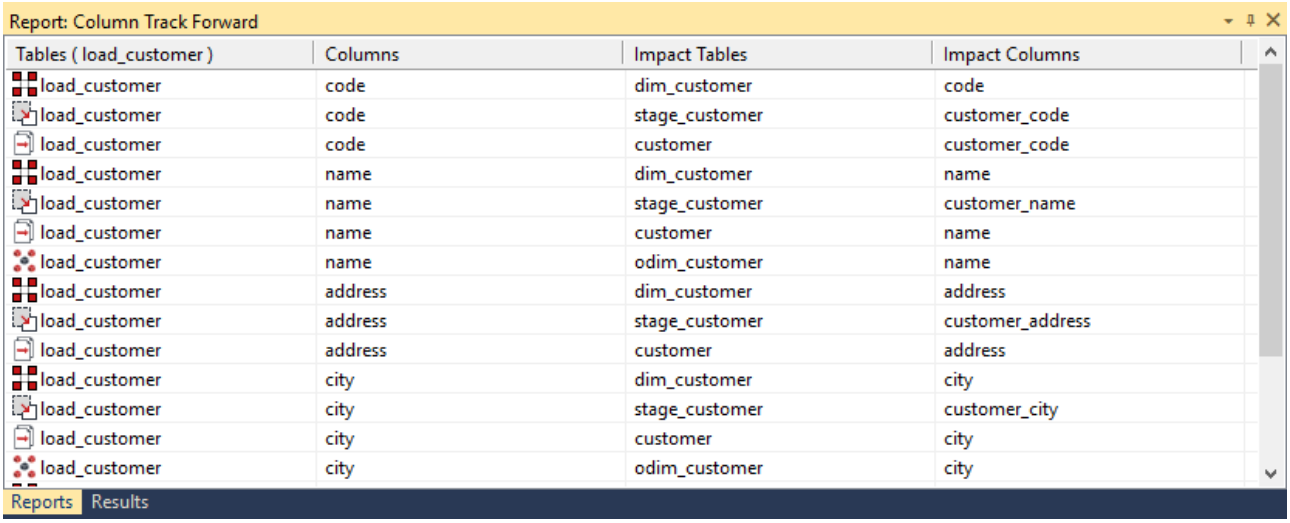
- Groups
- Projects
- Table
- Column

© Copyright, WhereScape Software Ltd All rights reserved.



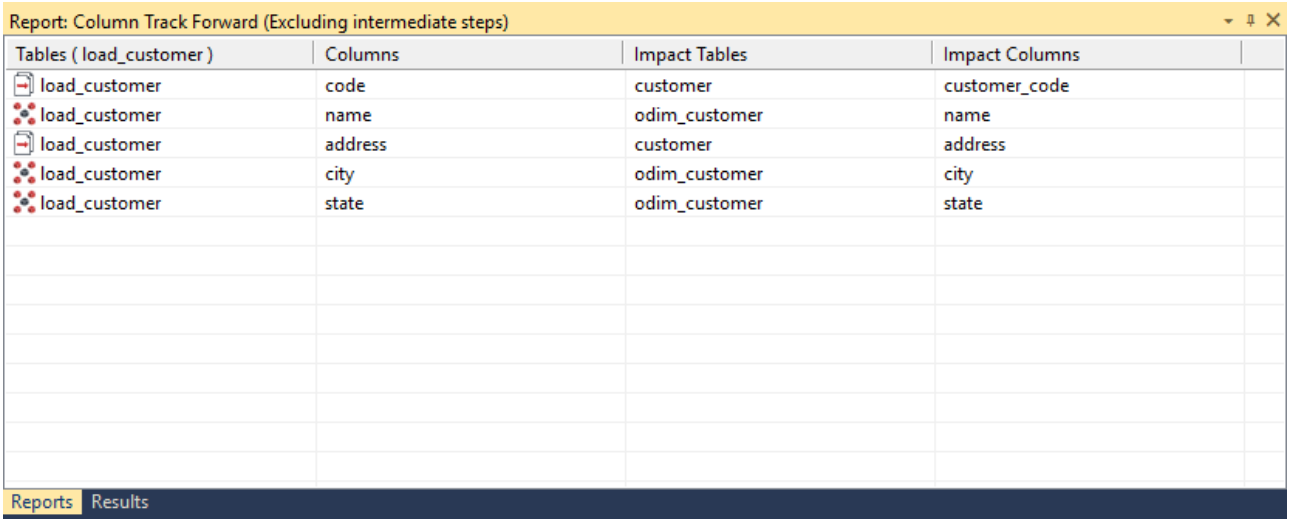
Results

If you left the **Exclude Intermediate Steps** checkbox **unchecked** then the results screen is as follows, showing the impacted tables for the selected tables:



Tables (load_customer)	Columns	Impact Tables	Impact Columns
load_customer	code	dim_customer	code
load_customer	code	stage_customer	customer_code
load_customer	code	customer	customer_code
load_customer	name	dim_customer	name
load_customer	name	stage_customer	customer_name
load_customer	name	customer	name
load_customer	name	odim_customer	name
load_customer	address	dim_customer	address
load_customer	address	stage_customer	customer_address
load_customer	address	customer	address
load_customer	city	dim_customer	city
load_customer	city	stage_customer	customer_city
load_customer	city	customer	city
load_customer	city	odim_customer	city

If however, you selected **Exclude Intermediate Steps** then the results screen is as follows, showing only the final impacted table for the selected tables:



Tables (load_customer)	Columns	Impact Tables	Impact Columns
load_customer	code	customer	customer_code
load_customer	name	odim_customer	name
load_customer	address	customer	address
load_customer	city	odim_customer	city
load_customer	state	odim_customer	state

The results of this report are displayed as a list of impacted tables and columns, the order of the result set determining the immediate impact. The report includes the following columns:

- Tables *(the names of selected tables)*
- Columns *(the names of selected columns)*

© Copyright, WhereScape Software Ltd All rights reserved.

- Impact Tables
- Impact Columns

The result set is **not** sortable, as the order of the result set determines the immediate lineage.

Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

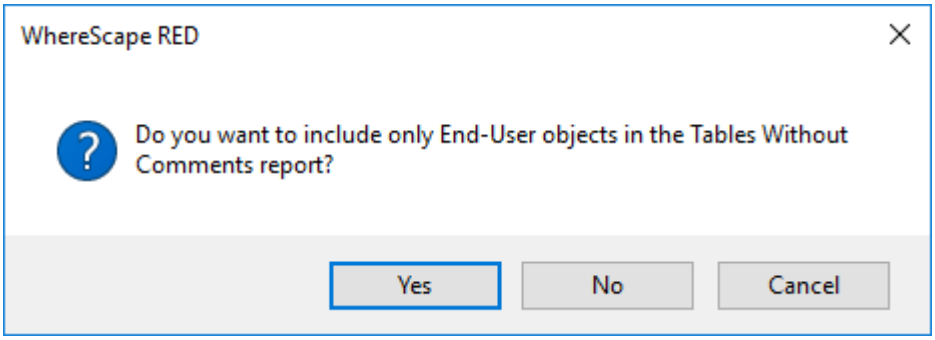
Table Reports

There are five reports for analyzing Tables:

- Tables Without Comments
- Load Tables by Connection
- Export Objects by Connection
- Records that Failed a Dimension Join
- External source Table/Files

Tables without Comments

This report shows **table** objects in the metadata that don't have descriptions. The report prompts the user with the option to only display user facing table objects or all table objects.



Objects Included

The following WhereScape RED object types are included in this report:

- All Tables objects or
- Only End User Layer (EUL) objects

Note

The EUL object are set in **Tools > Options > Object Types > Object Type End User Setting.**

Parameters

There are no parameters for this report.

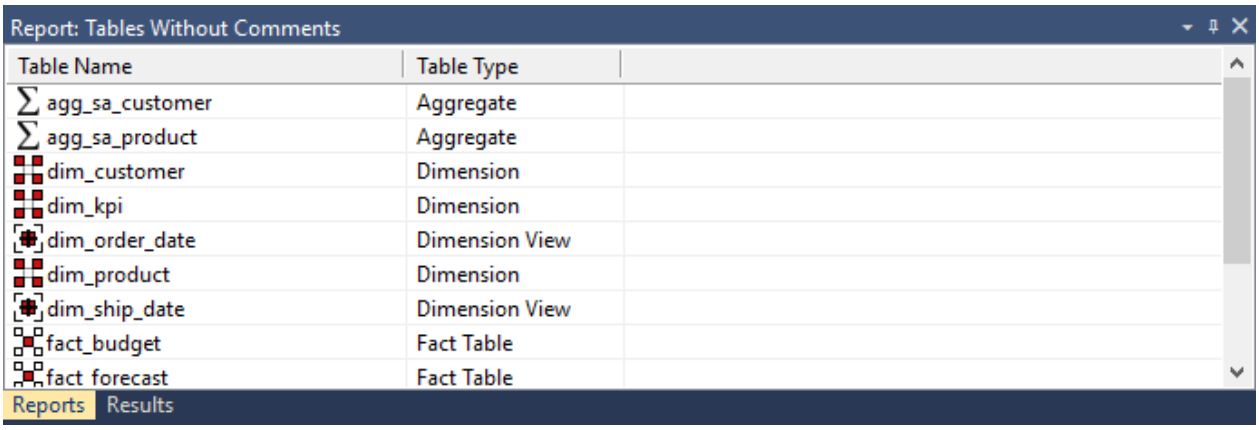
Results

The results of this report are displayed as a list of objects in the metadata missing comments with the following columns:

- Table name *(the name of the table)*
- Table type

© Copyright, WhereScape Software Ltd All rights reserved.

Report Example



Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Load Tables by Connection

This report shows load tables in the metadata repository with their Connection and Source schema or database.

Objects Included

The following WhereScape RED object types are included in this report:

- Load Tables

Parameters

There are no parameters for this report.

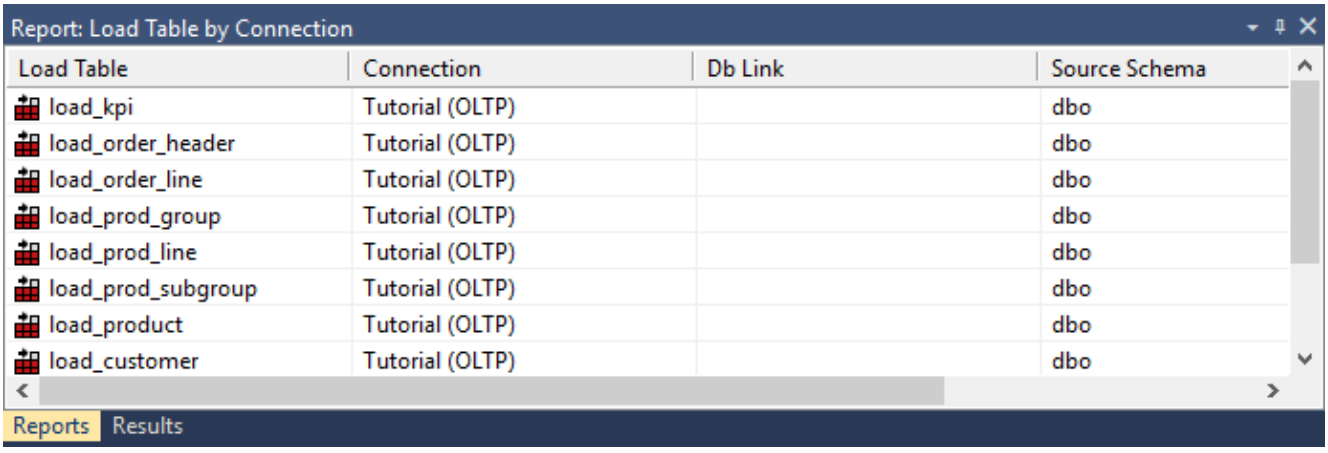
Results

The results of this report are displayed as a list of objects in the metadata repository with the following columns:

- Load table *(the name of the load table)*
- Connections
- Source Schema *(the name the database or schema the load table is source from - blank for files)*

The result set is sortable by clicking on the appropriate column heading.

Report Example



Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

© Copyright, WhereScape Software Ltd All rights reserved.

Export Objects by Connection

This report shows export tables in the metadata repository with their Connection and Source schema or database.

Objects Included

The following WhereScape RED object types are included in this report:

- Export Tables

Parameters

There are no parameters for this report.

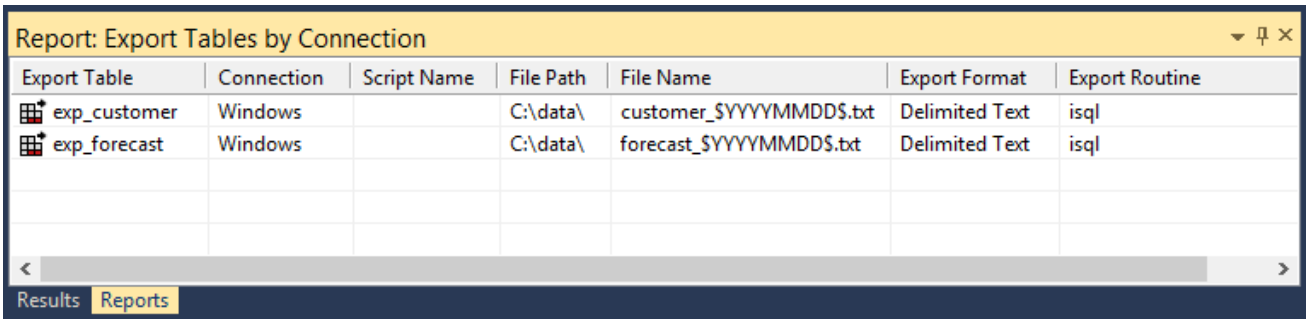
Results

The results of this report are displayed as a list of objects in the metadata repository with the following columns:

- Export Table (the name of the export table)
- Connection
- Script Name
- File Path
- File Name
- Export Format
- Export Routine

The result set is sortable by clicking on the appropriate column heading.

Report Example



Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Records that failed a Dimension Join

This report shows the dimension business key(s) that could not be found in a specified dimension when a specified staging table was last updated. This report will show null values, blank values and business keys not found in the dimension.

Objects Included

The following WhereScape RED object types are included in this report:

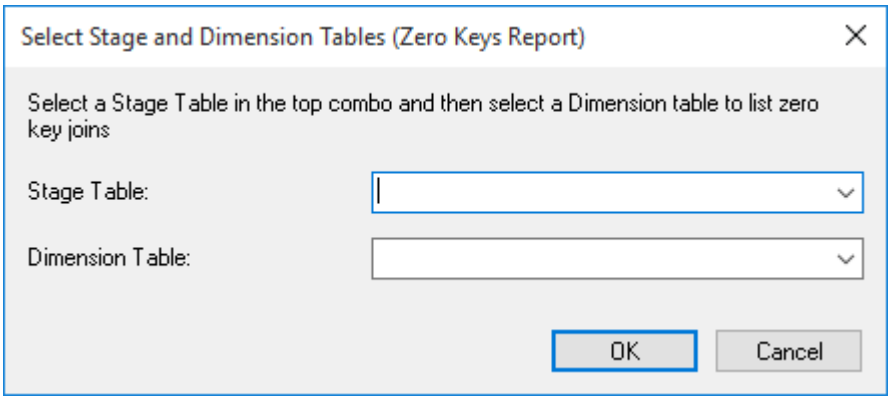
- Stage Tables

Parameters

This report requires two parameters to be specified:

- Stage Table Name *(the staging table to be checked)*
- Dimension Table Name *(the dimension table of the dimension key in the staging table selected first)*

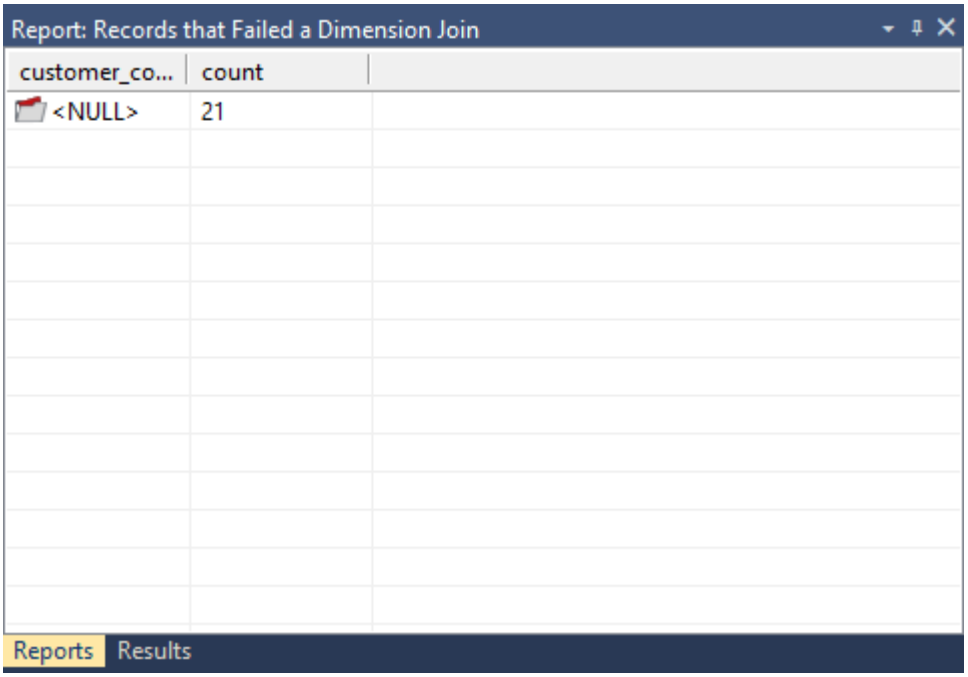
© Copyright, WhereScape Software Ltd All rights reserved.



Results

The report contains a list of values not found in the dimension and a count for each value. Each column is sortable by clicking on the appropriate column heading.

Report Example



Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

External Source Tables/Files

This report shows external sources for load tables in the metadata repository.

Objects Included

The following WhereScape RED object types are included in this report:

- Load Tables

Parameters

There are no parameters for this report.

Results

The results of this report are displayed as a list of objects in the metadata repository with the following columns:

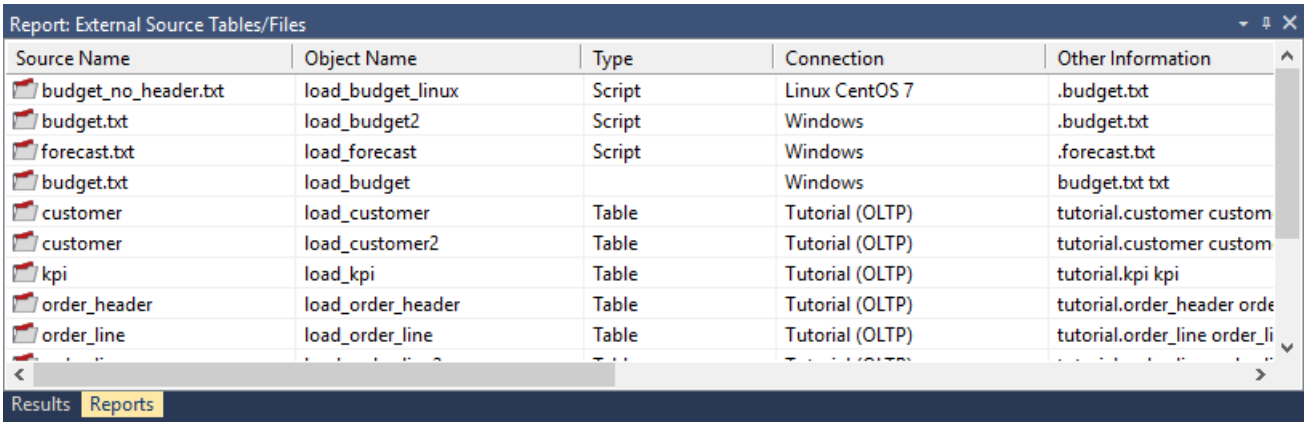
- Source name *(the name of the object's source)*
- Object name *(the name of the object)*

© Copyright, WhereScape Software Ltd All rights reserved.

- Type *(the type of object the source is: Table or File)*
- Connection
- Other information *(for tables, the source schema/database.source table; for files, the file name)*

The result set is sortable by clicking on the appropriate column heading.

Report Example



Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Procedure Reports

There are two reports for analyzing Procedures:

- Modified Procedures
- Custom Procedures

Modified Procedures

This report shows **modified** procedures in the metadata repository with their creation and modification dates.

Objects Included

The following WhereScape RED object types are included in this report:

- Procedures

Parameters

There are no parameters for this report.

Results

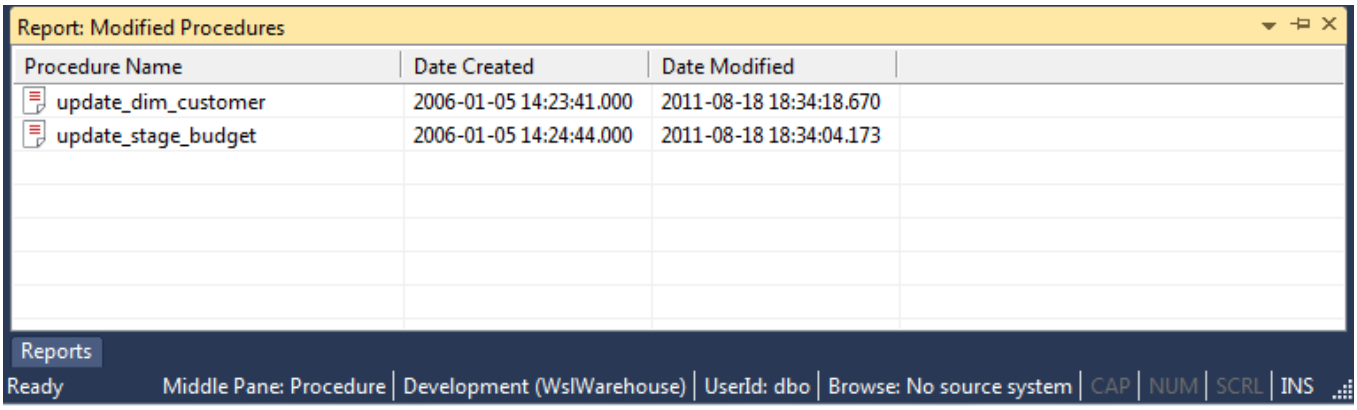
The results of this report are displayed as a list of modified procedures in the metadata repository with the following columns:

- Procedure Name *(the name of the object)*
- Dated Created
- Date Modified *(last modification date)*

The result set is sortable by clicking on the appropriate column heading.

© Copyright, WhereScape Software Ltd All rights reserved.

Report Example



Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Custom Procedures

This report shows **custom** procedures in the metadata repository with their creation and modification dates.

Note

Custom procedures are procedures generated for any table object as a **Custom Procedure**. Custom procedures are generated via a template.

Objects Included

The following WhereScape RED object types are included in this report:

- Procedures

Parameters

There are no parameters for this report.

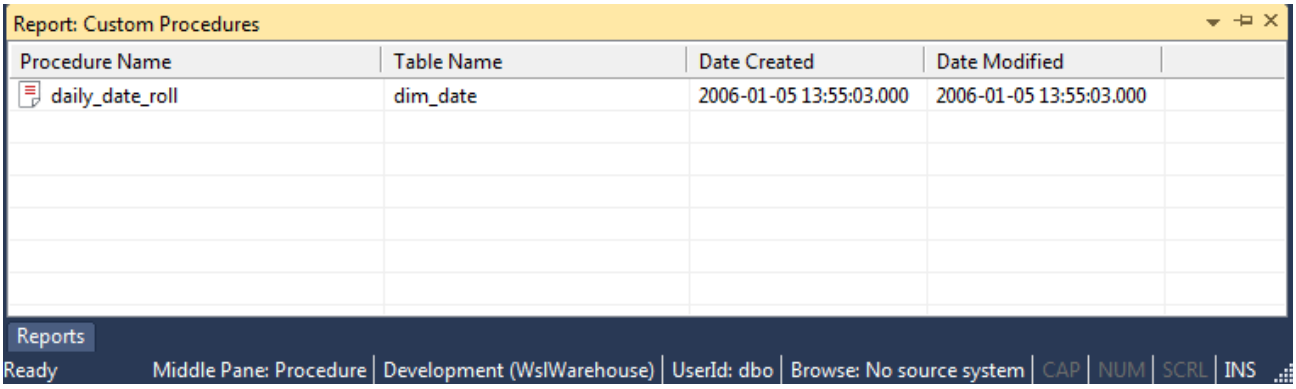
Results

The results of this report are displayed as a list of custom procedures in the metadata repository with the following columns:

- Procedure Name *(the name of the object)*
- Table Name *(the table object the procedure is attached to)*
- Dated Created
- Date Modified *(last modification date)*

The result set is sortable by clicking on the appropriate column heading.

Report Example



Sending Results to Microsoft Excel

Right-click the result set and select **Output to File** to send the results of this report to Microsoft Excel.

© Copyright, WhereScape Software Ltd All rights reserved.

Index Reports

There is one report for analyzing Indexes:

- Modified Indexes

Modified Indexes

This report shows indexes in the metadata repository with their creation and modification dates.

Objects Included

The following WhereScape RED object types are included in this report:

- Indexes

Parameters

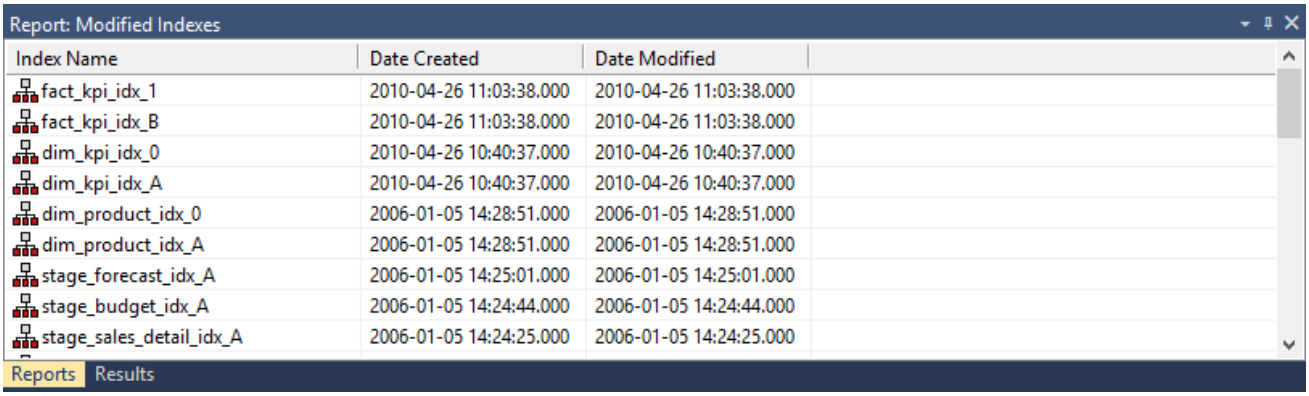
There are no parameters for this report.

Results

- The results of this report are displayed as a list of indexes in the metadata repository with the following columns:
- Index Name *(the name of the index)*
- Dated Created
- Date Modified *(last modification date)*

The result set is sortable by clicking on the appropriate column heading.

Report Example



Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Object Reports

There are five reports for analyzing Objects:

- Objects-Project Matrix
- Modified Objects (excluding indexes)
- Objects Checked-out
- Loaded or Imported Objects
- Objects with Extended Properties

Objects-Project Matrix

This report lists the objects that exist in projects other than **All Objects**.

Objects Included

All object types are included in this report.

Parameters

There are no parameters for this report.

© Copyright, WhereScape Software Ltd All rights reserved.

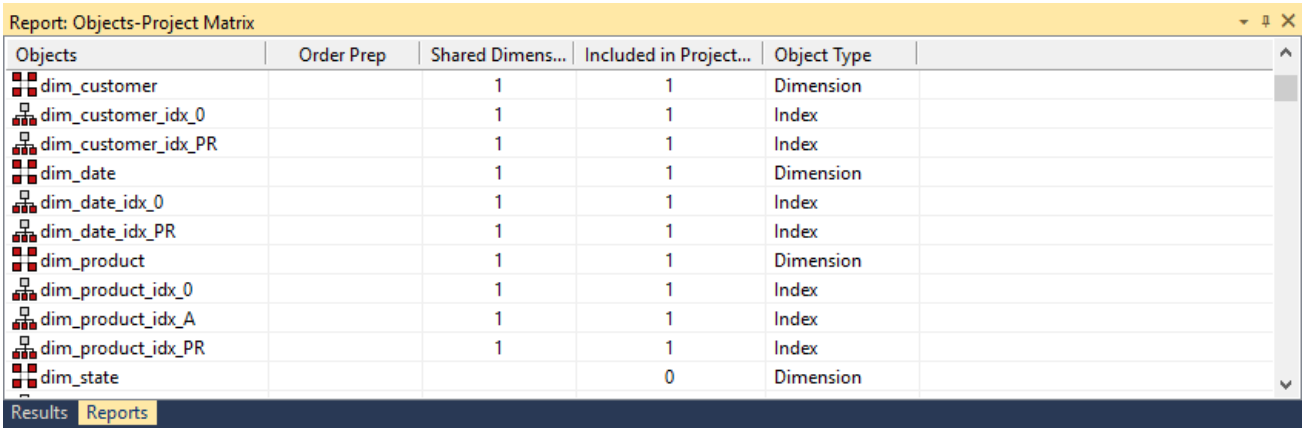
Results

The results of this report are displayed as a list of objects in the metadata repository that are in one or more projects (other than All Objects) with the following columns as a grid:

- Objects *(the name of the object)*
- Project Name 1 *(heading is the name of the first project, value is a 1 to indicate the object is in this project, blank otherwise)*
- Project Name 2 *(heading is the name of the second project, value is a 1 to indicate the object is in this project, blank otherwise)*
- ...
- Project Name n *(heading is the name of the nth project, value is a 1 to indicate the object is in this project, blank otherwise)*

The result set is sortable by clicking on the appropriate column heading.

Report Example



Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Modified Objects (excluding indexes)

This report shows objects in the metadata repository with their creation and modification dates.

Objects Included

The following WhereScape RED object types are included in this report:

- Load Tables
- Stage Tables
- Dimension Tables and Views
- Fact Tables
- Aggregate Tables
- Exports
- Procedures
- Host Scripts

Parameters

There are no parameters for this report.

Results

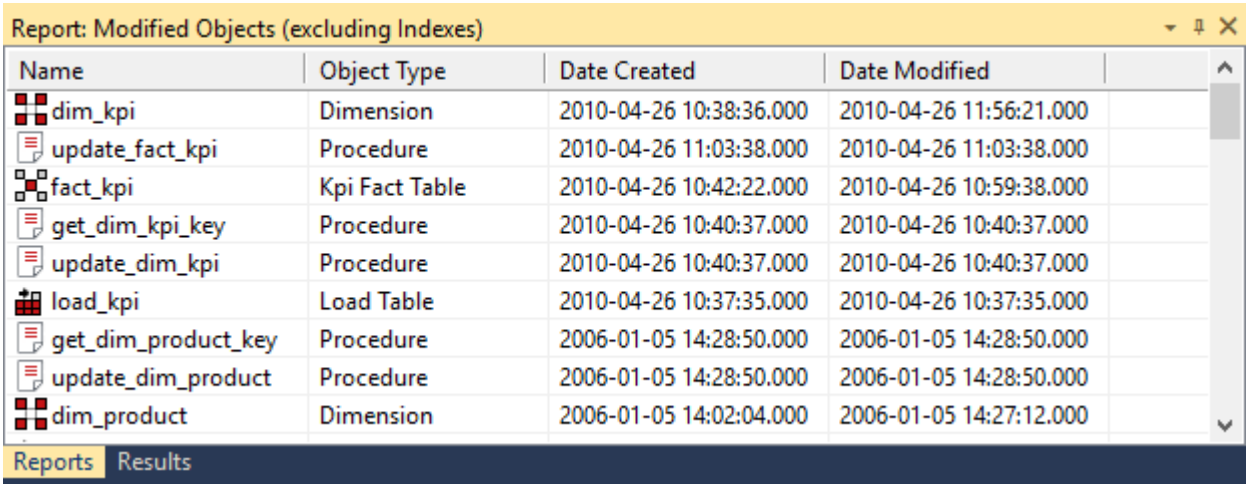
The results of this report are displayed as a list of objects in the metadata repository with the following columns:

- Name *(the name of the object)*
- Object Type
- Dated Created
- Date Modified *(last modification date)*

© Copyright, WhereScape Software Ltd All rights reserved.

The result set is sortable by clicking on the appropriate column heading.

Report Example



Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Objects Checked-out

This report lists all objects currently checked out.

Objects Included

All object types and data warehouse tables can be included in this report.

Parameters

There are no parameters for this report.

Results

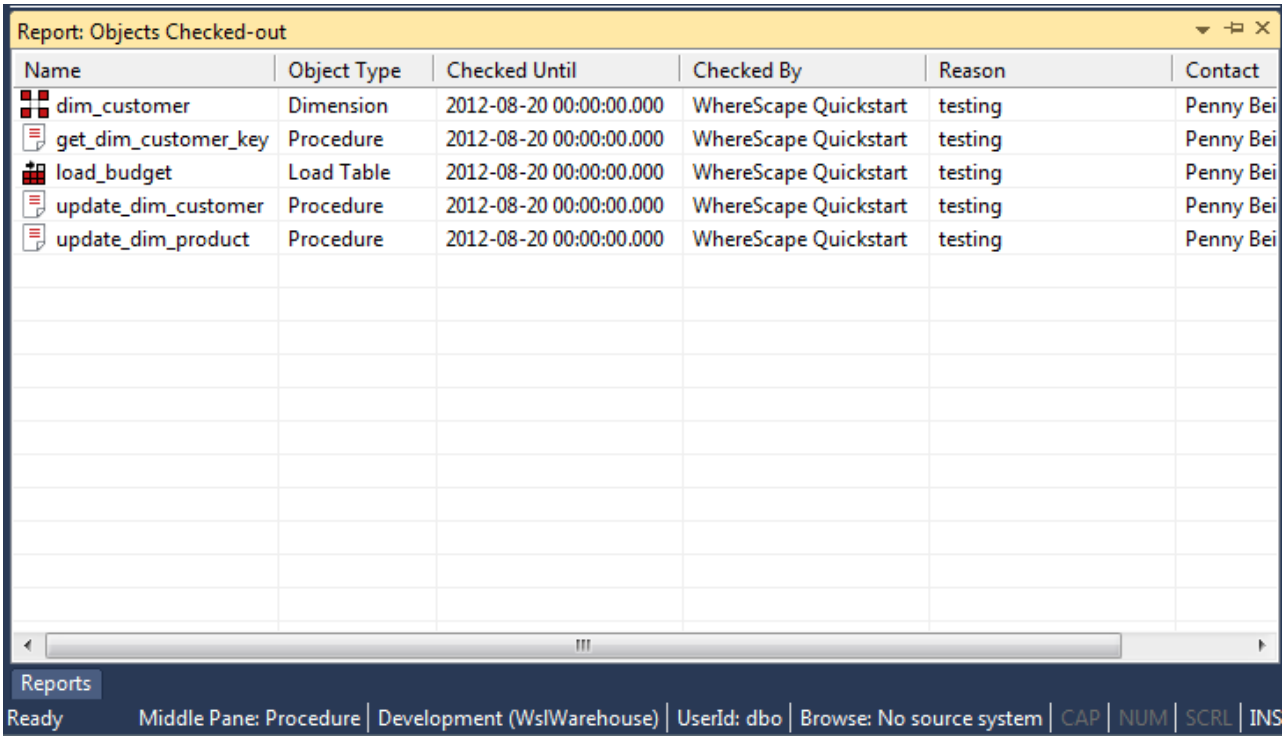
The results of this report are displayed with the following columns:

- Name *(The name of the object)*
- Object Type *(The type of the object, e.g. Fact, Dimension, etc.)*
- Checked Until *(The date the object will be automatically checked back in)*
- Checked By *(The name of the WhereScape RED user who checked out the object)*
- Reason *(The reason provided for checking out the object)*
- Contact *(The contact details provided when the object was checked out)*

The result set is sortable by clicking on the appropriate column heading.

© Copyright, WhereScape Software Ltd All rights reserved.

Report Example



Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Loaded or Imported Objects

This report lists objects in the metadata repository that have been refreshed or imported from another repository.

Objects Included

The following WhereScape RED object types are included in this report:

- Load Tables
- Stage Tables
- Dimension Tables and Views
- Fact Tables
- Aggregate Tables
- Join Indexes
- Views
- Cubes
- Exports
- Procedures
- Host Scripts

Notes

- Indexes are not included.
- Objects created via Application Load must have Auto Version enabled to be included in the report.

© Copyright, WhereScape Software Ltd All rights reserved.

Parameters

There are no parameters for this report.

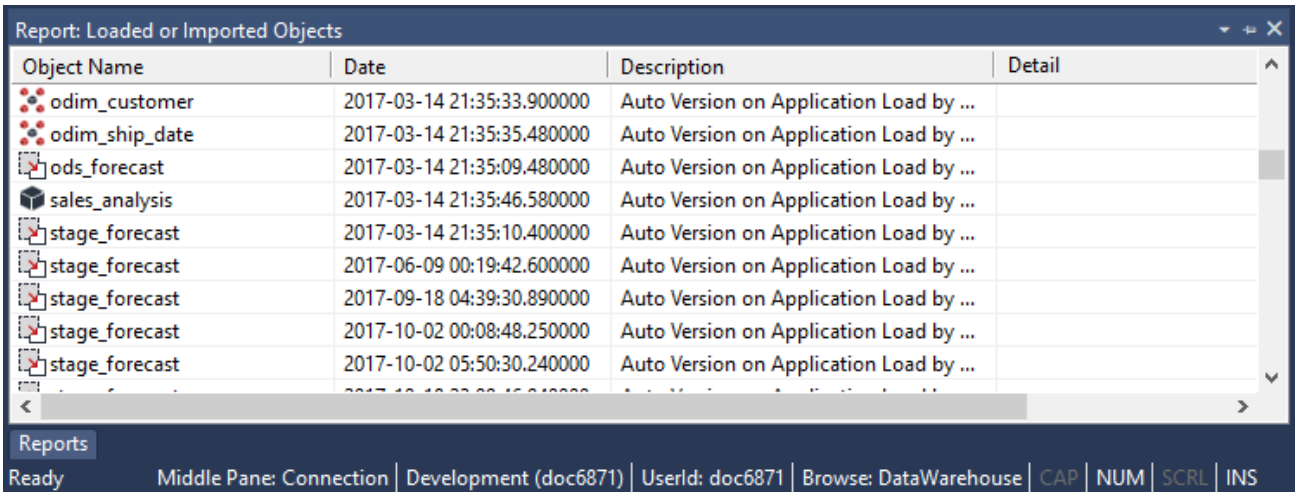
Results

The results of this report are displayed as a list of refreshed objects in the metadata repository with the following columns:

- Object Name
- Date *(of last refresh or import)*
- Description *(the kind of import or refresh)*
- Detail *(not currently used)*

The result set is sortable by clicking on the appropriate column heading.

Report Example



Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Objects with Extended Properties

This report lists objects in the metadata repository with Extended Properties defined.

Objects Included

The following WhereScape RED object types are included in this report:

- Connections
- Load Tables
- Stage Tables
- Data Store Tables
- Hub Tables
- Link Tables
- Satellite Tables
- EDW 3NF Tables
- Dimension Tables and Views
- Fact Tables
- Aggregate Tables
- Join Indexes
- Views
- Exports

© Copyright, WhereScape Software Ltd All rights reserved.

Parameters

There are no parameters for this report.

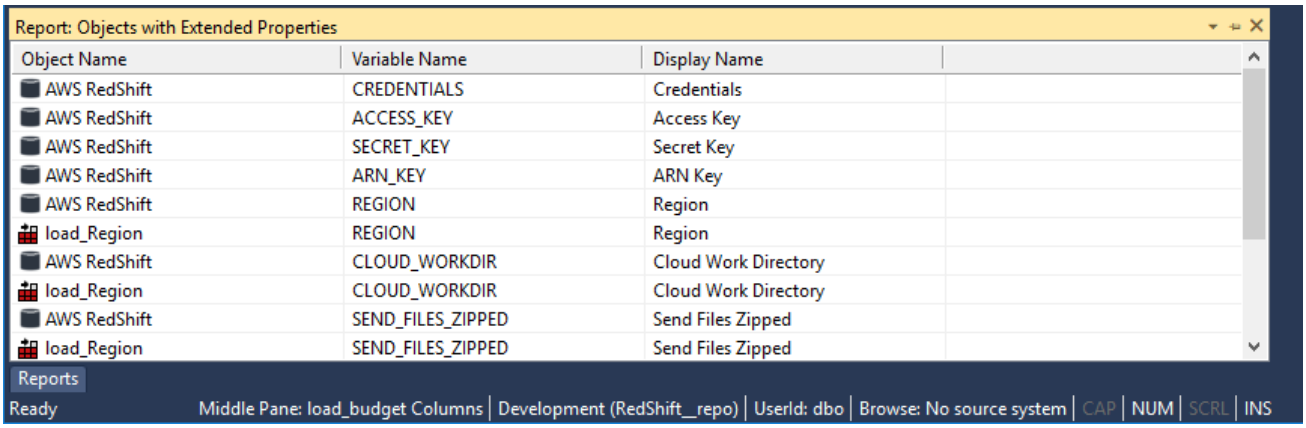
Results

The results of this report are displayed as a list of refreshed objects in the metadata repository with the following columns:

- Object Name
- Variable Name *(The unique variable name in the format it is accessed from the RED metadata service and included into scripts)*
- Display Name *(The extended property name displayed in the Connection or Table properties)*

The result set is sortable by clicking on the appropriate column heading.

Report Example



Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Job Reports

There are three reports for analyzing Jobs:

- Object by Scheduler Job
- Jobs with an Object
- Tasks of a Job

Objects by Scheduler Job

This report shows all jobs and objects as well as the object actions. Table and Cube objects not in any jobs are displayed with a job name of **No Job**.

Objects Included

All object types are included in this report.

Parameters

There are no parameters for this report.

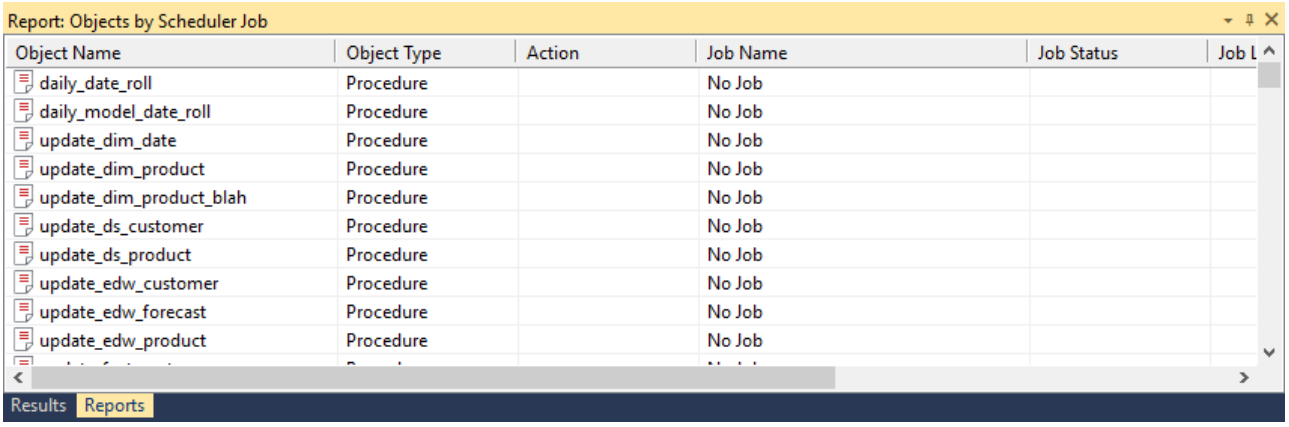
Results

The results of this report are displayed as a list of jobs with the following columns:

- Table Name *(the name of the table or cube)*
- Action
- Job Name *(the name of the job)*
- Job Status
- Job Last Run
- Job Next Run

© Copyright, WhereScape Software Ltd All rights reserved.

Report Example



Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Jobs with an Object

This report shows all jobs containing a selected object and lists the job action.

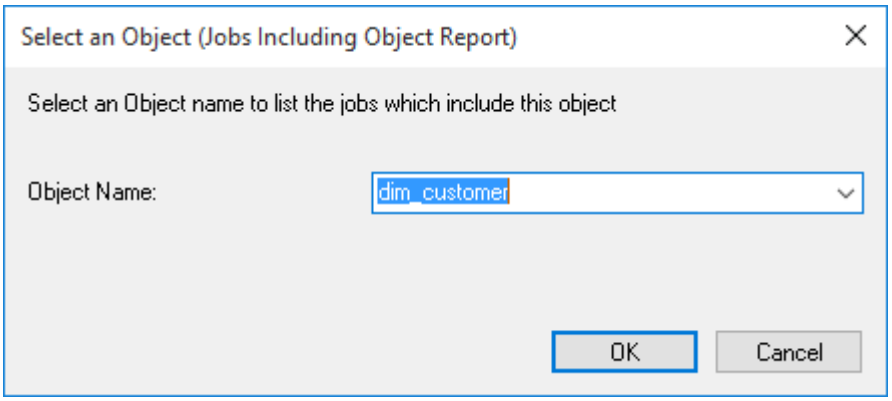
Objects Included

All object types are included in this report.

Parameters

This report has one parameter:

- Object Name



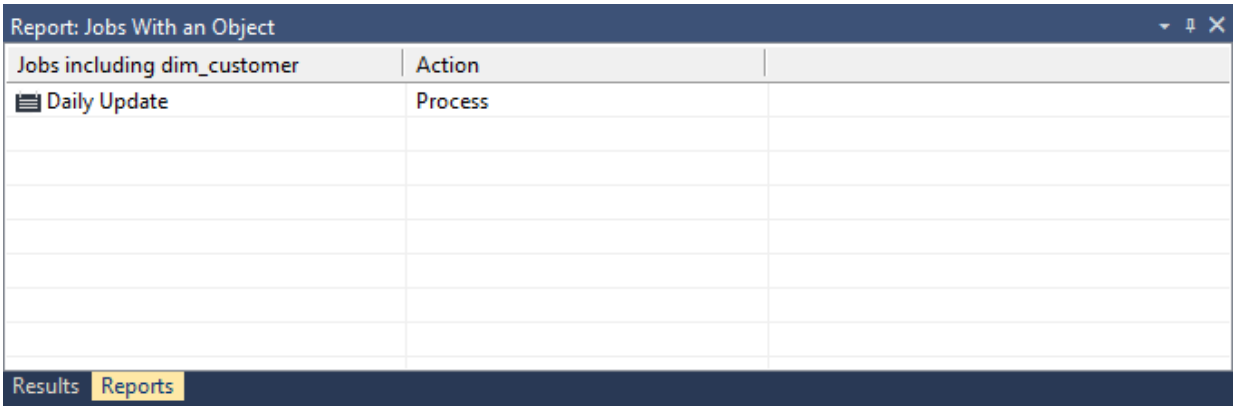
Results

The results of this report are displayed as a list of jobs with the following columns:

- Jobs including **object_name**
- Action

© Copyright, WhereScape Software Ltd All rights reserved.

Report Example



Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Tasks of a Job

This report shows all tasks for a selected job including dependencies.

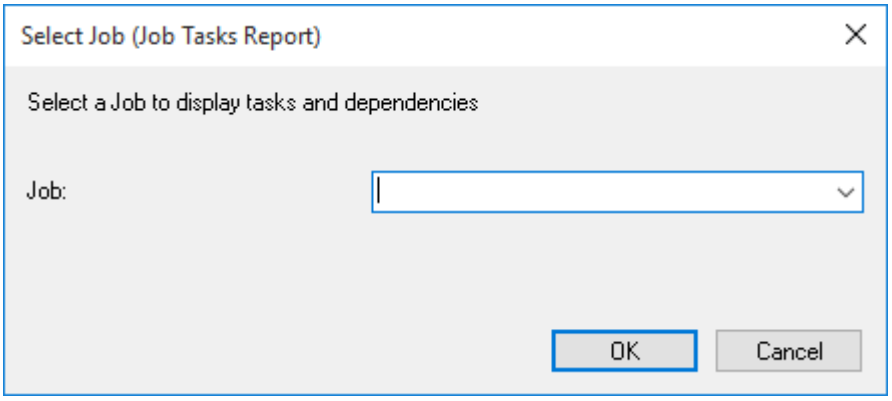
Objects Included

All object types are included in this report.

Parameters

This report has one parameter:

- Job Name



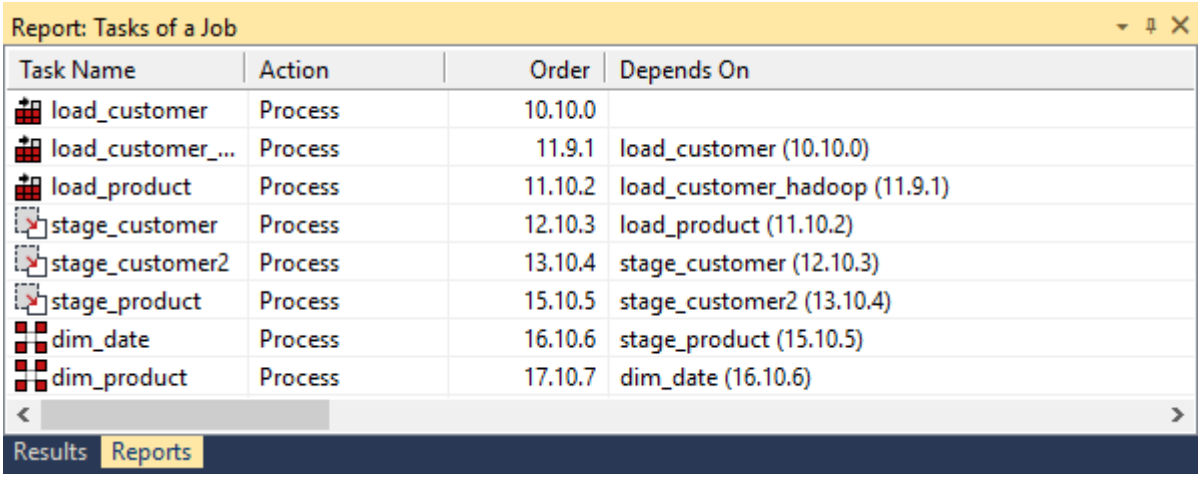
Results

The results of this report are displayed as a list of task dependencies with the following columns:

- Task name *(the table, Index, procedure or script name)*
- Action
- Order *(the order number as shown in the edit tasks dialog in the scheduler)*
- Depends On *(the task(s) and order number this task depends on)*

© Copyright, WhereScape Software Ltd All rights reserved.

Report Example



Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Operational Reports

There are four Operational Reports:

- Object Performance History
- Job Performance History
- Task Performance History
- Fragmentation

Object Performance History

This report shows the audit trail from the scheduler logs for a selected object.

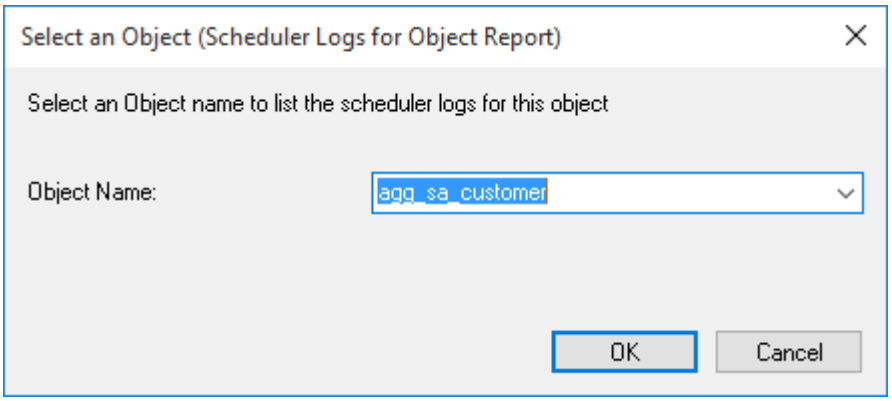
Objects Included

All object types are included in this report.

Parameters

This report has one parameter:

- Object Name



Results

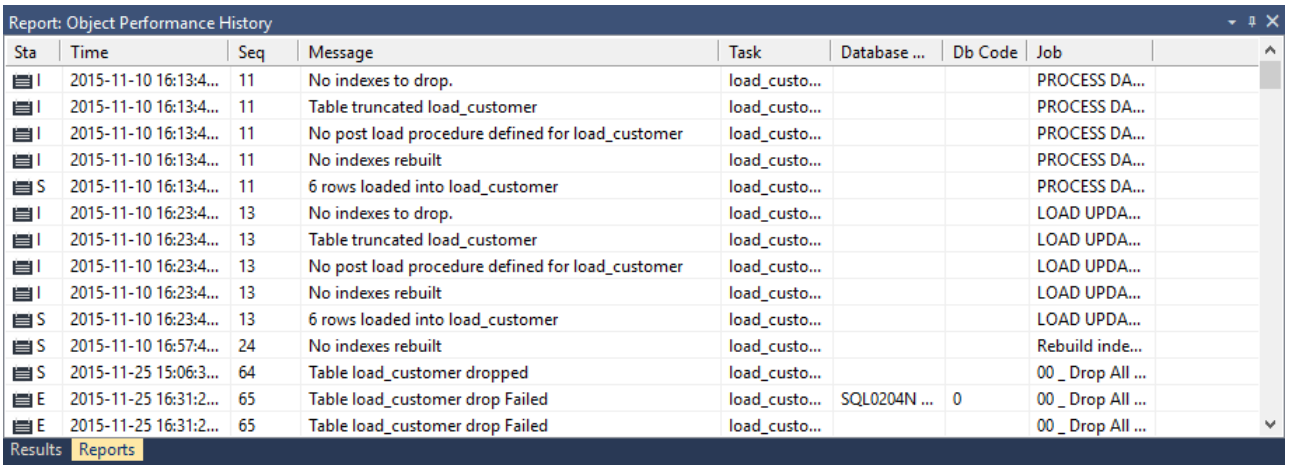
The results of this report are displayed as a list of audit log entries with the following columns:

- Sta *(the type of audit log entry)*
- Time *(the date and time the audit log entry was written)*
- Seq *(the job sequence of the job writing the audit log entry)*
- Message *(the message in the Audit log)*

© Copyright, WhereScape Software Ltd All rights reserved.

- Task *(the object name)*
- Job *(the name of the job that ran the task)*

Report Example



Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Job Performance History

This report shows the performance (duration) over time of a selected job.

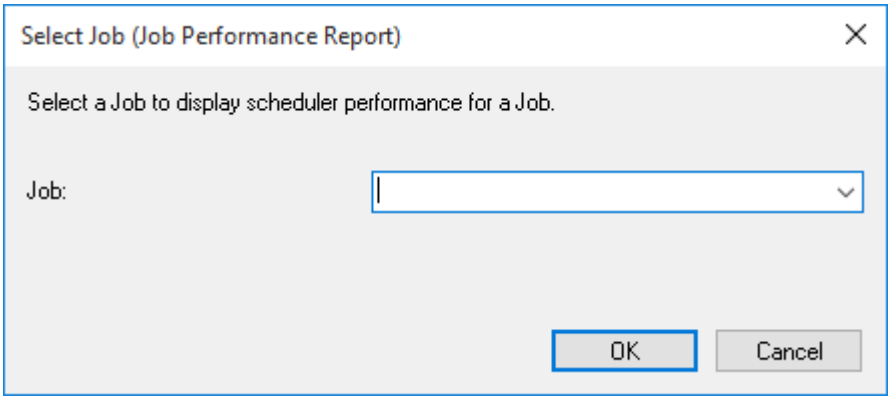
Objects Included

All object types are included in this report.

Parameters

This report has one parameter:

- Job Name



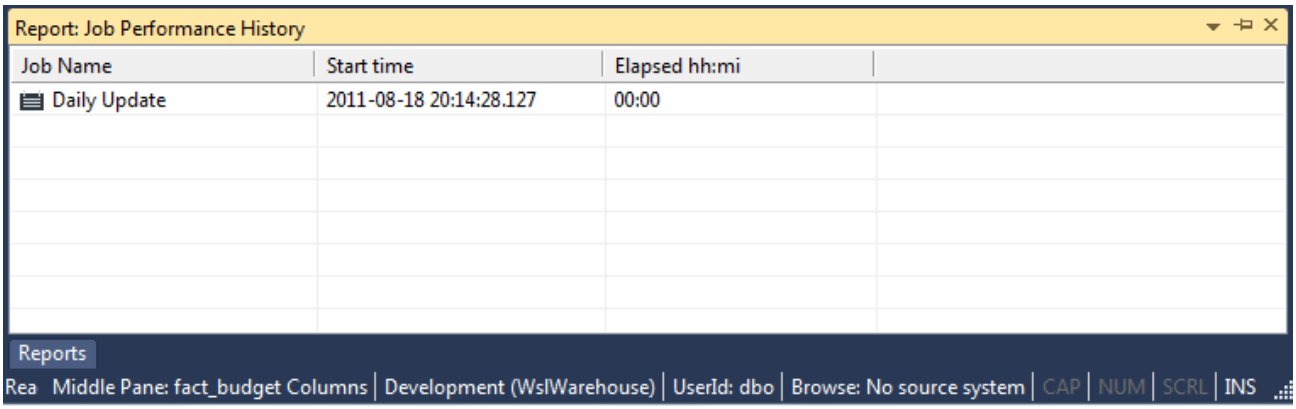
Results

The results of this report are displayed as a list of job instances with the following columns:

- Job name *(the name of the job)*
- Start time *(the date and time the job started)*
- Elapsed hh:mm *(the duration of the job)*

© Copyright, WhereScape Software Ltd All rights reserved.

Report Example



Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Task Performance History

This report shows the performance (duration) over time of a selected task within a selected job.

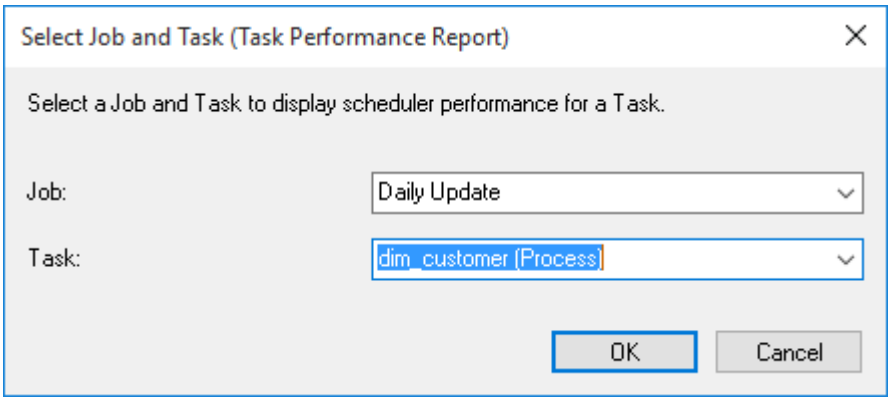
Objects Included

All object types are included in this report.

Parameters

This report has two parameters:

- Job Name
- Task Name (including action)



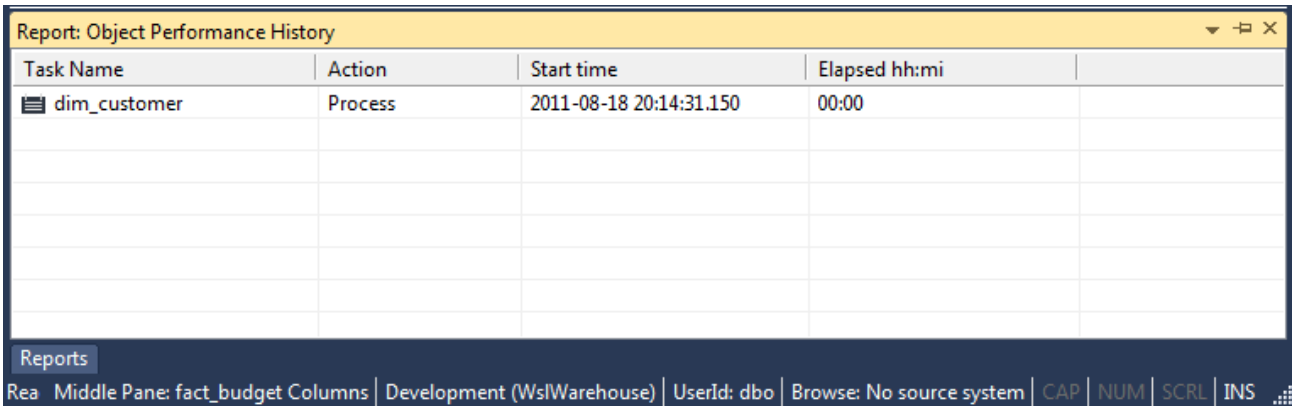
Results

The results of this report are displayed as a list of task instances for the selected job with the following columns:

- Task name (*the table, Index, procedure or script name)*
- Action
- Start time *(the date and time the task started)*
- Elapsed hh:mm *(the duration of the task)*

© Copyright, WhereScape Software Ltd All rights reserved.

Report Example



Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Fragmentation

This report is only available on a SQL Server database.

Objects Included

All object types are included in this report.

Parameters

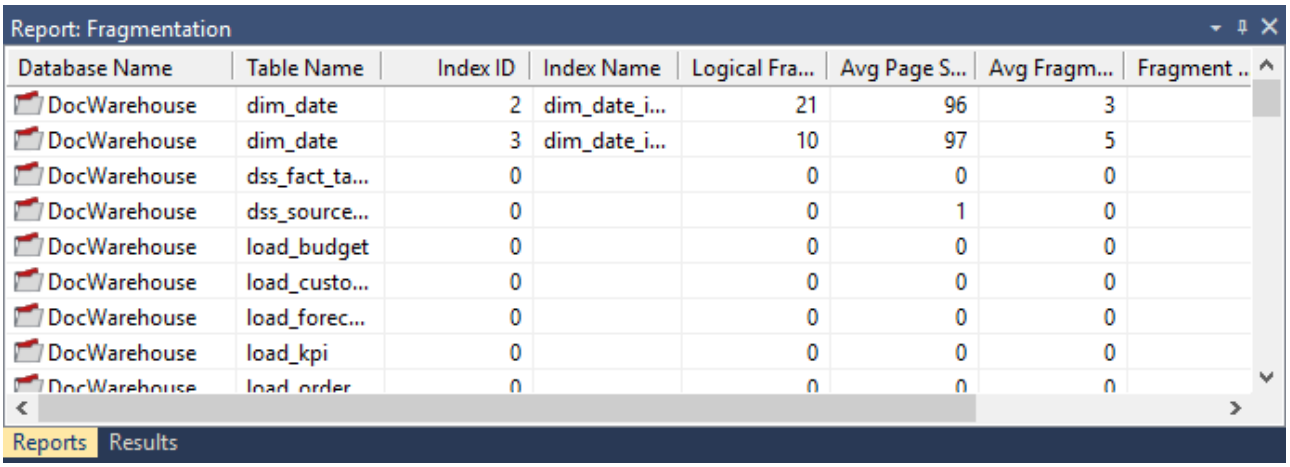
This report has no parameters.

Results

The results of this report are displayed as a list of tables within the database with the following columns:

- Database
- Table Name
- Index ID
- Index Name
- Logical Fragmentation
- Avg Page Space Used
- Avg Fragment Size in Pages
- Fragment Count
- Page Count
- Size in GB

Report Example



Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

© Copyright, WhereScape Software Ltd All rights reserved.

Validate

When these Validate processes are run, the results are displayed in the middle pane of the RED screen; the results of the other reports are displayed in a separate tab in the bottom pane of the RED screen.

Validate Meta-data

This process validates the Meta data. If problems are encountered, the results are displayed in the middle pane.

Use the right-click option against each identified issue to apply a repair.

Sending Results to Microsoft Excel

Right-click on the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Validate Workflow Data

This process validates the Workflow data. If problems are encountered, the results are displayed in the middle pane.

Use the right-click option against each identified issue to apply a repair.

Sending Results to Microsoft Excel

Right-click on the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Validate Table Create Status

This process validates a table structure in the meta data against the table in the database.

Select one or more tables and click the **Validate Selected** button or click the **Validate All** button to validate all the tables.

1. If a table is found to be different then it can be altered by using the right-click menu option when positioned over the table name.
2. If the update date and the modified in database date imply a change that is not apparent then these dates can be re-synced in the same way.

Sync Column order with database

Right click on the result set and select **Sync Column order with database** to reorder the metadata columns to match the column order in the database table.

Sending Results to Microsoft Excel

Right-click on the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Validate Load Table Status

This process compares a load table in the meta data with the table in the source system. It compares the columns and column data types.

Sending Results to Microsoft Excel

Right-click on the result set and click **Output to File** to send the results of this report to Microsoft Excel.

© Copyright, WhereScape Software Ltd All rights reserved.

Validate Index Status

This process validates an index structure in the meta data against the index in the database. Select one or more indexes and click the **Validate Selected** button or click the **Validate All** button to validate all indexes.

Sending Results to Microsoft Excel

Right-click on the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Validate Procedure Status

This process compares a procedure in the meta data with the compiled version of the same procedure stored within the database. The subsequent display will report either a match or a difference.

If a procedure is found to differ then you can use the procedure editor to examine the exact differences by selecting the **Tools > Compare to Compiled Source** option.

Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

List Metadata Tables not in the Database

This report shows database **table** objects in the metadata that don't exist in the data warehouse database.

Objects Included

The following WhereScape RED object types are included in this report:

- Load Tables
- Stage Tables
- Dimension Tables and Views
- Fact Tables
- Aggregate Tables
- Retro Copies (but not Retro Definitions)

Parameters

There are no parameters for this report.

Results

The results of this report are displayed as a list of objects in the metadata not in the data warehouse database.

Report Example

© Copyright, WhereScape Software Ltd All rights reserved.

Sending Results to Microsoft Excel

Right-click on the result set and click **Output to File** to send the results of this report to Microsoft Excel.

List Database Tables not in the Meta-data

This report shows database **table** objects that exist in the data warehouse database but not in the metadata.

Objects Included

The following WhereScape RED object types are included in this report:

- Load Tables
- Stage Tables
- Dimension Tables and Views
- Fact Tables
- Aggregate Tables
- Retro Copies (but not Retro Definitions)

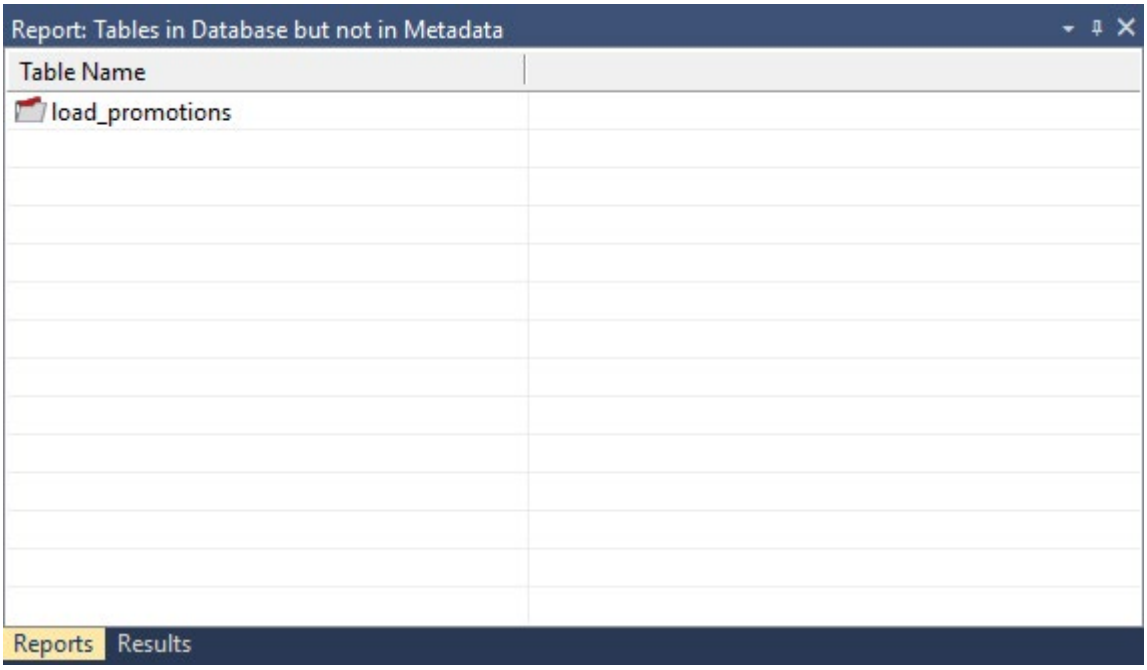
Parameters

There are no parameters for this report.

Results

The results of this report are displayed as a list of objects in the data warehouse database not in the metadata.

Report Example



Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

© Copyright, WhereScape Software Ltd All rights reserved.

List Tables with no related Procedures or Scripts

This report shows all table objects (certain types of objects only - see below) in the metadata repository that don't have an associated update procedure.

Objects Included

The following WhereScape RED object types are included in this report:

- Load Tables **(script based loads only)**
- Stage Tables
- Dimension Tables and Views
- Fact Tables
- Aggregate Tables
- Data Store Tables
- EDW 3NF Tables

Parameters

There are no parameters for this report.

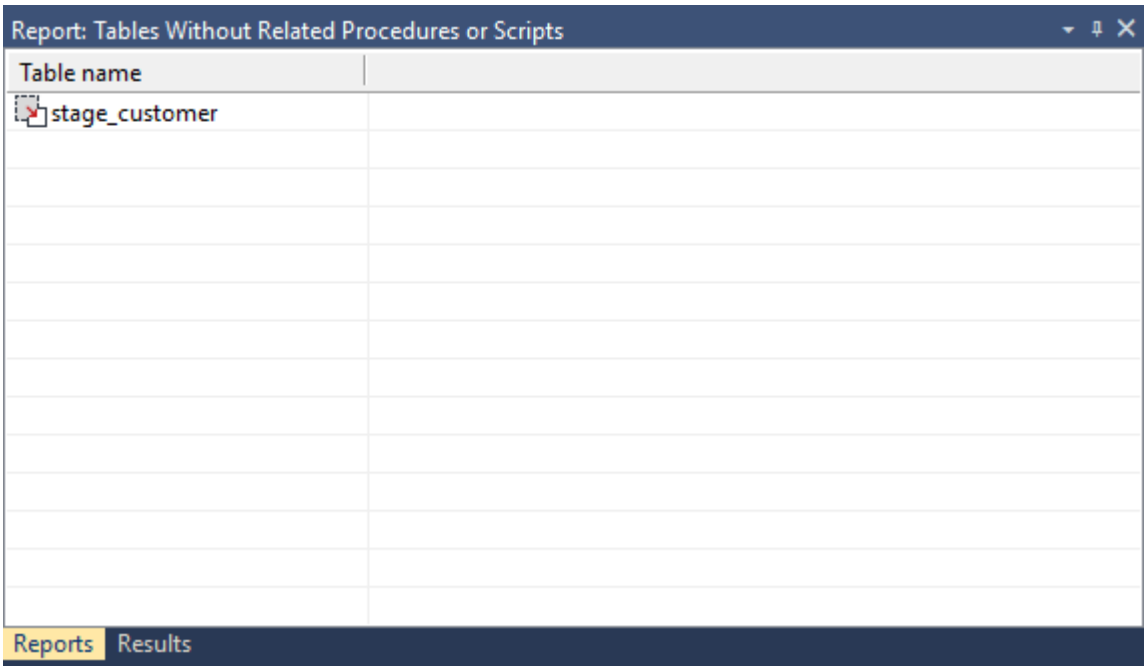
Results

The results of this report are displayed as a list of table objects in the metadata repository with the following columns:

- Table name

The result set is sortable by clicking on the appropriate column heading.

Report Example



Sending Results to Microsoft Excel

Right-click on the result set and click **Output to File** to send the results of this report to Microsoft Excel.

© Copyright, WhereScape Software Ltd All rights reserved.

List Procedures not related to a Table

This report shows all procedures and host scripts in the metadata repository that aren't associated with a table object.

Objects Included

The following WhereScape RED object types are included in this report:

- Procedures
- Host Scripts

Parameters

There are no parameters for this report.

Results

The results of this report are displayed as a list of code objects in the metadata repository with the following columns:

- Procedure/Script name

The result set is sortable by clicking on the appropriate column heading.

Report Example

Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

© Copyright, WhereScape Software Ltd All rights reserved.

Compare Meta-data Repository to Another

This report compares the current metadata repository to a remote repository.

Objects Included

All object types are included in this report.

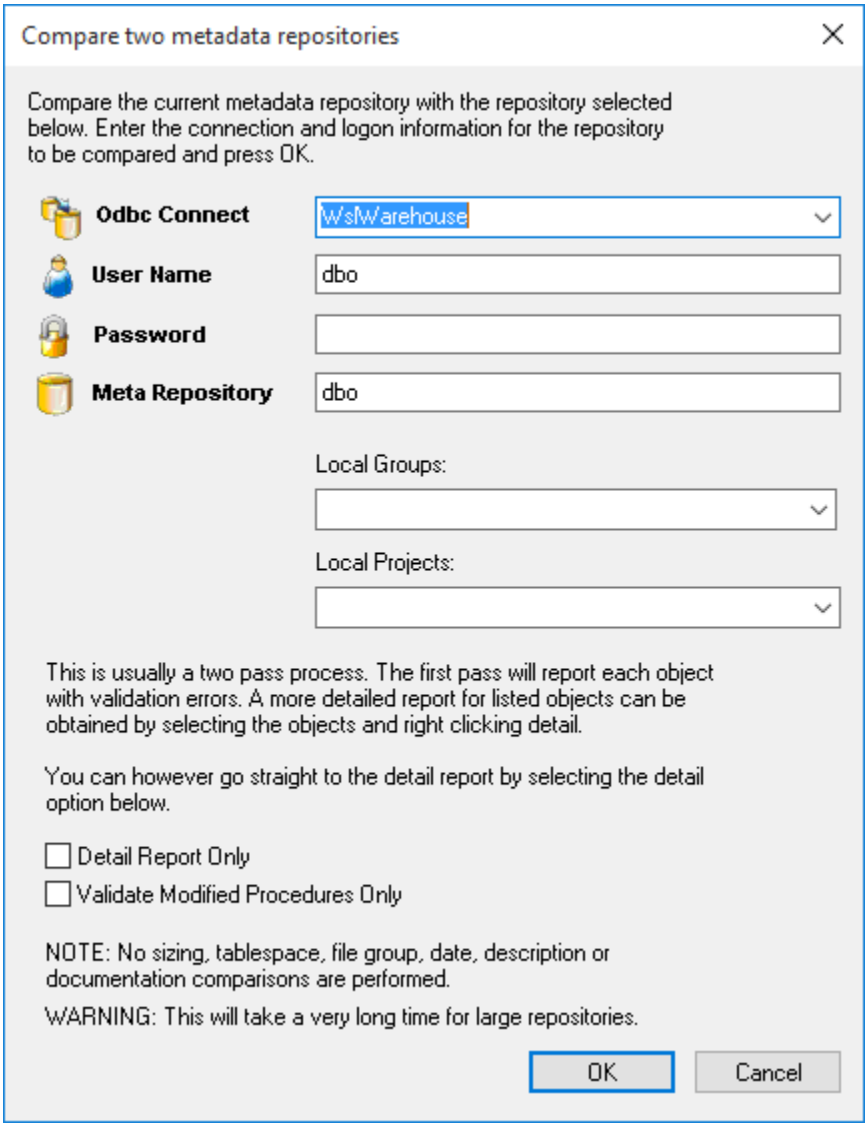
Parameters

This report requires connection information for the remote repository to be entered. Specifically:

- ODBC Connect **(the ODBC source for the other repository)**
- User Name **(user name for the remote repository)**
- Password **(password for the remote repository)**
- Meta Repository **(database/user the remote repository metadata is stored in)**

Four additional parameters can be specified:

- Optional filter on Local Groups
- Optional filter on Local Projects
- Do detail report
- Only validate procedures that have been modified



© Copyright, WhereScape Software Ltd All rights reserved.

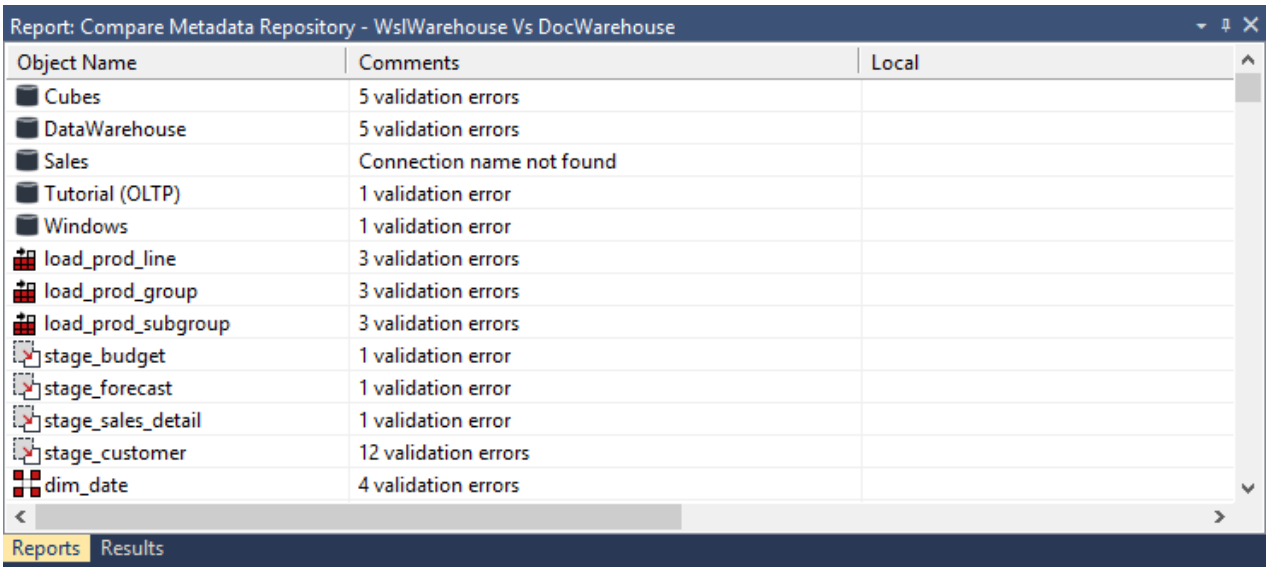
Results

The results of this report are displayed as a list of differences with the following columns:

- Object Name
- Comments **(Summary difference between current and selected repository)**
- Local
- Remote

The result set is sortable by clicking the appropriate column heading.

Report Example



Checking result details:

- Right-click the object name with validation errors.
- Select **Detail**.
- This will rerun the report just for the selected object and will display more details about the errors in the Comments, Local and/or Remote Column(s).

Sending Results to Microsoft Excel

- Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Compare Meta-data Indexes to Database

This report shows database indexes in the data warehouse database that don't exist in the metadata.

Objects Included

The following WhereScape RED object types are included in this report:

- Indexes

Parameters

There are no parameters for this report.

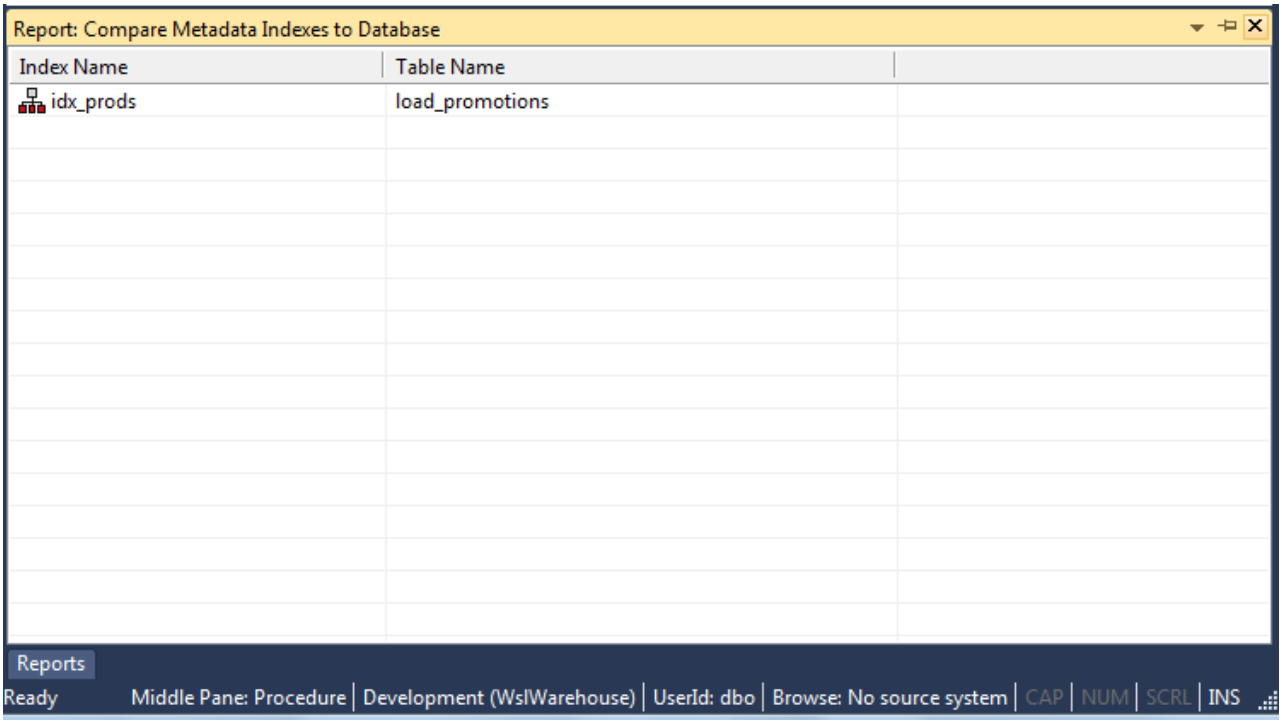
© Copyright, WhereScape Software Ltd All rights reserved.

Results

The results of this report are displayed as a list of indexes in the data warehouse database not in the metadata with the following columns:

- Index Name **(the name of the index)**
- Table Name

Report Example



Sending Results to Microsoft Excel

- Right-click on the result set and click **Output to File** to send the results of this report to Microsoft Excel.

List Duplicate Business Key Columns

This report shows any columns that are the business (natural) key of more than one table.

Objects Included

The following WhereScape RED object types are included in this report:

- Stage Tables
- Dimension Tables and Views
- Fact Tables
- Aggregate Tables

Parameters

There are no parameters for this report.

Results

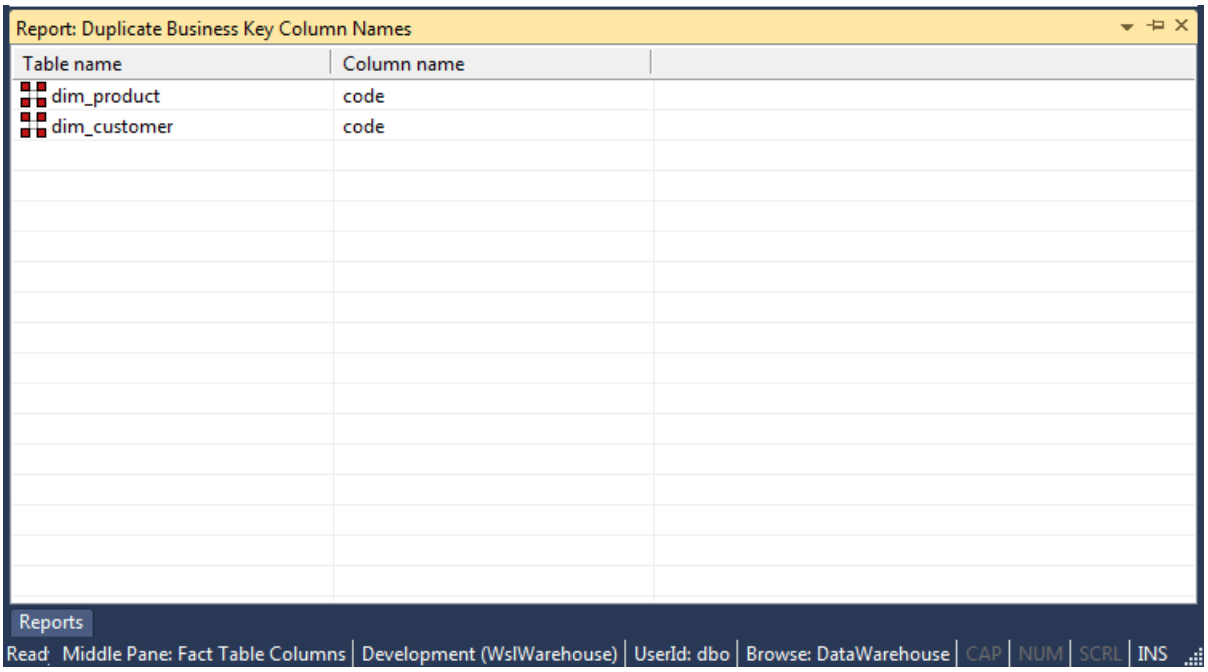
The results of this report are displayed as a list of table columns with the following columns:

- Table name
- Column name

© Copyright, WhereScape Software Ltd All rights reserved.

The result set is sortable by clicking the appropriate column heading.

Report Example



Sending Results to Microsoft Excel

- Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Query Data Warehouse Objects

This report allows SQL queries to be run as the user signed into the repository.

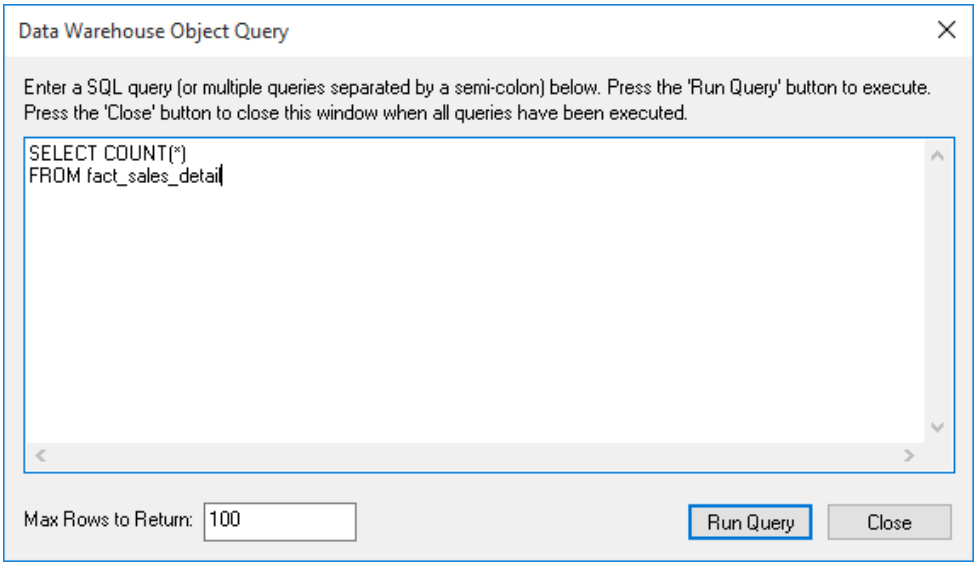
Objects Included

All object types and data warehouse tables can be included in this report.

Parameters

This report has one parameter:

- The SQL Query



© Copyright, WhereScape Software Ltd All rights reserved.

Results

The results of this report are displayed with the following columns:

- First SQL column SELECTed
- Second SQL column SELECTed
- ...
- nth SQL column SELECTed

The result set is sortable by clicking on the appropriate column heading.

Report Example

Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

© Copyright, WhereScape Software Ltd All rights reserved.

Promoting Between Environments

This chapter covers the promotion of metadata objects between environments. Various methods exist for getting new or changed metadata from one repository into another repository.
It is of course possible and in fact desirable to have multiple metadata repositories. At the very least we would normally have a development and a production environment.
In some situations, it may also be desirable to have multiple child development repositories with one master repository where all elements are brought together. WhereScape RED supports this type of structure but does not include source control or co-ordination of the repositories. It is up to the data warehouse manager to manually ensure that the various objects are kept in sync and coordinated.
As with any software system there are issues around how code is moved from a development environment into a testing or production environment.
This promotion of objects can be achieved via a number of different methods. Each is discussed below.
In summary, they are:

1. Updating a repository with an application or application patch.
2. Importing objects from another repository.
3. Restoring a full metadata set into a repository. Refer to **Backing Up and Restoring Metadata** for details.

Applications

The definition of an application is discussed in the following section on applications and the loading and updating of applications is discussed at some length in the RED Installation Guide. Only the concepts of the use of applications will be covered here.

An application is defined for our purposes as a group of objects. An application is a method of loading objects into a metadata repository. It can be used to upgrade or provide patches to an existing metadata repository. As such, an application can be used to distribute and remotely maintain a specific data warehousing solution. An application consists of a series of Windows files, which can be distributed to remote sites.
A list of the applications that have been applied to the metadata repository can be displayed via the **Tools > List Loaded Deployment Applications** menu option.

An application is created through the **Tools > Build Deployment Application** menu option. The created application can then update a metadata repository through the WhereScape RED Setup Administrator utility. In this manner, the application model can be used to update a metadata repository in an ordered and controlled fashion. Loading an application inserts various objects into the chosen metadata repository. An application is best defined as a set of objects that are shipped to enable inclusion of those objects in a remote repository.

Notes:

An application can only be loaded into a metadata repository running on the same database type as that of the application creator. (e.g. an Oracle application cannot be loaded into a SQL Server metadata repository).

WhereScape RED also enable you to create deployment applications via the **Projects/Groups and Object Groups** context menu, as well as from a **Diagram** generated in RED.

Application Creation

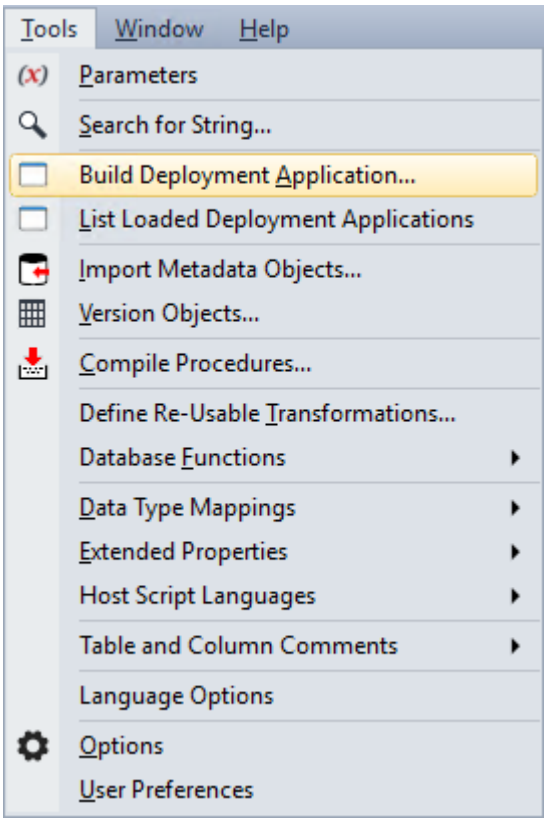
Tip:

You can automatically create an Application from:

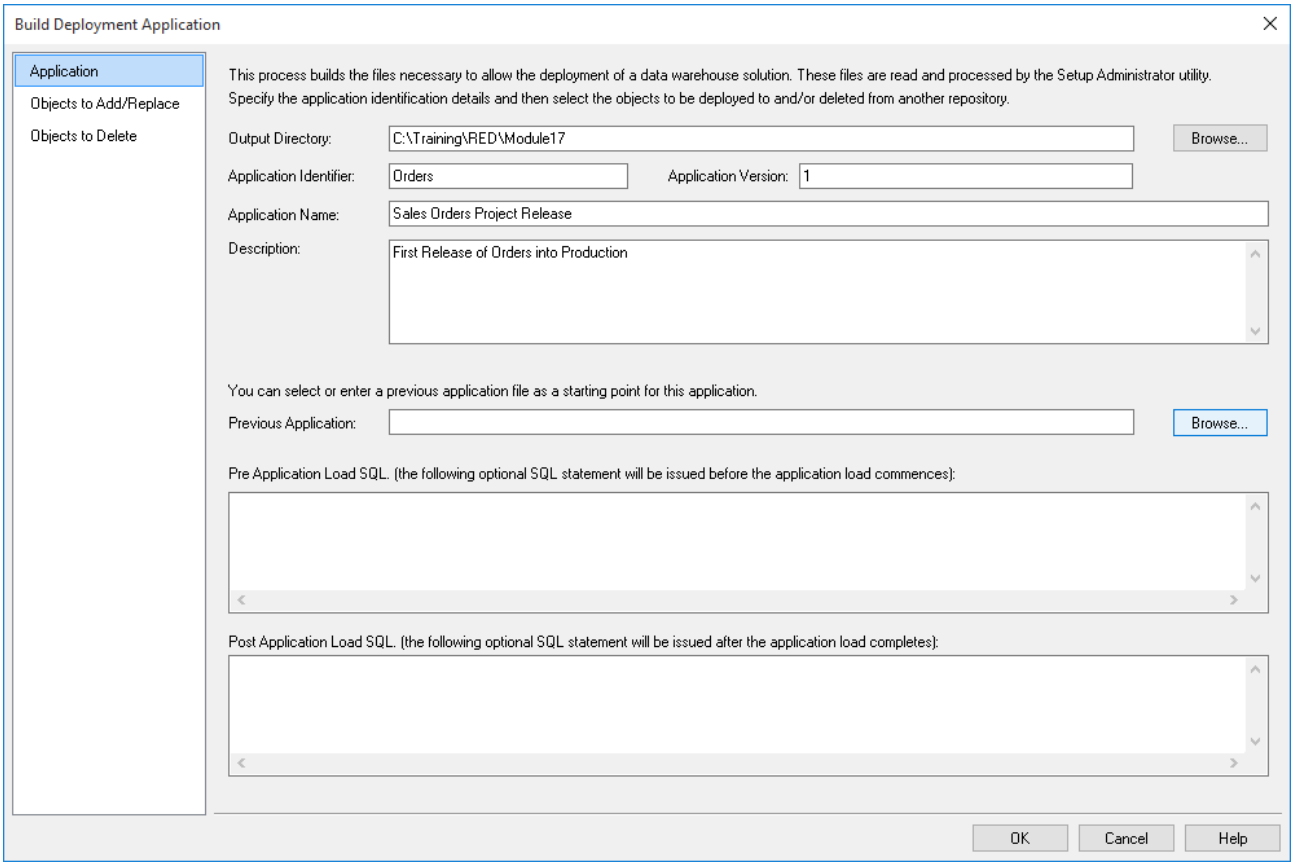
- Projects or Project Groups—refer to **Building an Application from Projects or Groups** for details.
- Object Groups—refer to **Building an Application from Object Groups** for details.
- Diagrams—refer to **Creating an Application from a Diagram** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

An application is created by selecting the **Tools > Build Deployment Application** menu option.



The following screen is displayed:



Once the application is defined and the objects selected, the application files are generated when you click the **OK** button.

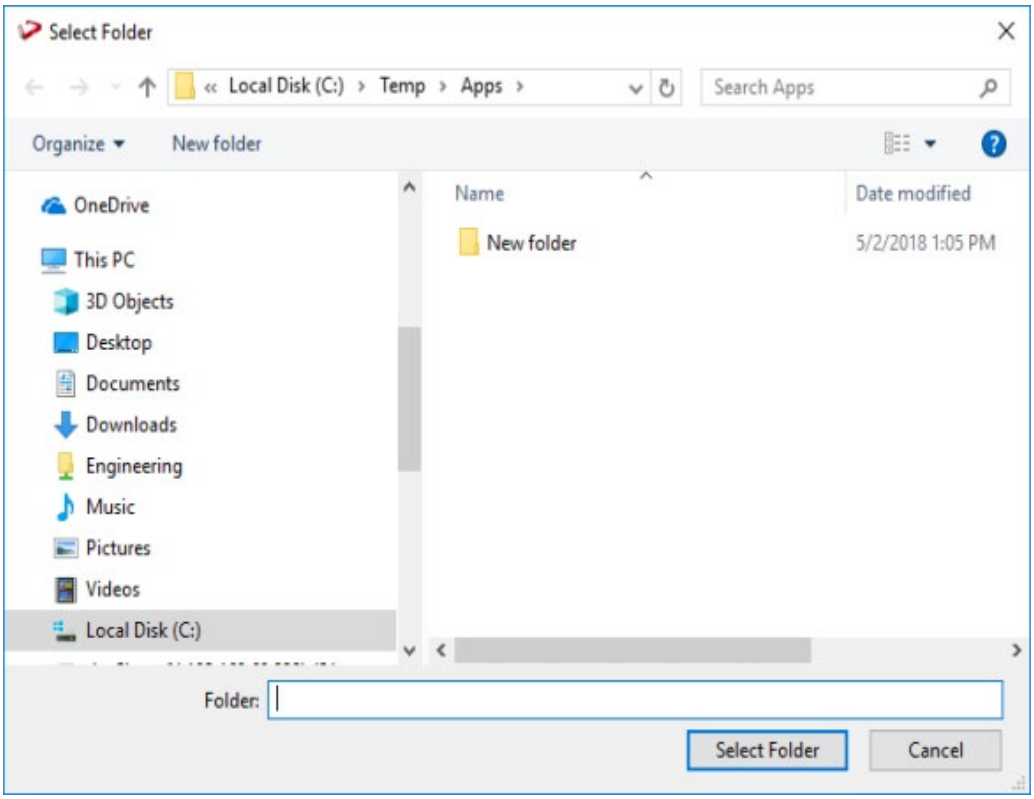
© Copyright, WhereScape Software Ltd All rights reserved.

If procedures are compiled as part of the subsequent application load, the compiles occur in the order in which they are listed in the application. If procedure dependencies exist, ensure that their ordering in the application object list is correct.
There are three tabs in the **Build Deployment Application** screen.

- The first tab defines the application.
- The second tab lists the objects to add or replace in the destination repository.
- The third tab lists the objects to delete in the destination repository.

Define an Application distribution

Application

Fields	Description
Output Directory	The directory to which the application files will be written. By default, this is the WhereScape program directory. **Note** To browse for the required folder, click the **Browse...** button. Click **New Folder** to create a new folder in the selected directory. 
Application Identifier	The application identifier is a four character code used to uniquely identify the application. This identifier is used in the naming of the files that are created to contain the application data.
Application Version	The version is a character string that provides a version number for reference purposes. This version number is displayed when applications are being loaded and is used in the naming of the files that are created to contain the application data. As such, it must contain characters that are valid in a Windows file name
Application Name	The name by which the application is known. This name is displayed during the selection of an application to load and is recorded in the metadata of the repository into which an application is loaded. It is not used, apart from documentation purposes.
Description	This description is displayed during the choosing of an application to load. It is not used at any other point, apart from documentation purposes.
Application	When an application is created the following files are built, where XXXX is the application

© Copyright, WhereScape Software Ltd All rights reserved.

<table>
<tr><th>Fields</th><th>Description</th></tr>
<tr><td>Files</td><td>identifier and NNNNN is the application version.
<table>
<tr><th>File</th><th>Purpose</th></tr>
<tr><td>App_data_XXXX_NNNNN.wst</td><td>This file contains the scripts and data required to rebuild the objects in the new metadata repository.</td></tr>
<tr><td>App_id_XXXX_NNNNN.wst</td><td>This control file identifies the application and its version.</td></tr>
<tr><td>App_obj_XXXX_NNNNN.wst</td><td>This file contains control information and each object in the application.</td></tr>
<tr><td>App_con_XXXX_NNNNN.wst</td><td>A list of all the connections either in the application or used by objects in the application.</td></tr>
<tr><td>App_map_XXXX_NNNNN.wst</td><td>A list of all the project and group mappings for the objects in the application.</td></tr>
</table></td></tr>
<tr><td>Previous Application</td><td>Click the Browse button next to Previous Application to choose a previously built application to use as a list of objects to include in the new application. After using a previous application as a starting point for this application, additional objects can be added or removed from the application.</td></tr>
<tr><td>Pre application load SQL</td><td>This text box enables you to enter an SQL statement that is executed before the application is loaded. For example, you may wish to drop the date dimension before loading the application because we have changed the primary key constraint. In such a case, you would enter 'drop table dim_date' in this field to have the table dropped before the application is loaded.</td></tr>
<tr><td>Post application load SQL</td><td>This text box enables you to enter an SQL statement that is executed after the application is loaded. For example, you could execute a function to populate a table.</td></tr>
</table>

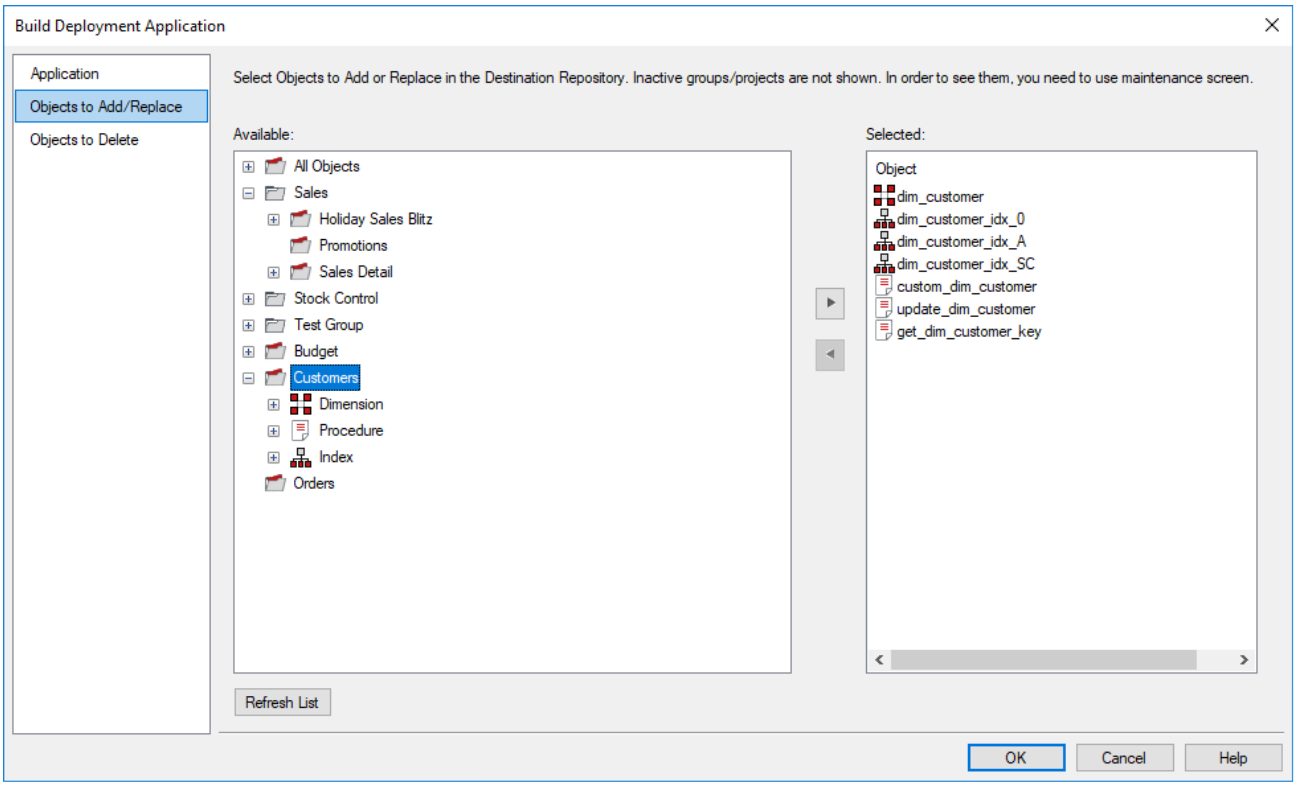
© Copyright, WhereScape Software Ltd All rights reserved.

Objects to Add/Replace

This tab enables you to select the objects to add or replace in the destination repository. Double click a Project/Group folder to display the objects that are associated with them. Objects can be moved from the left object tree pane to the right pane by double-clicking an object or by using the **>** button.

Note

Maximum number of objects in an application
5000 objects (including jobs)
2000 source views of views
1000 jobs.



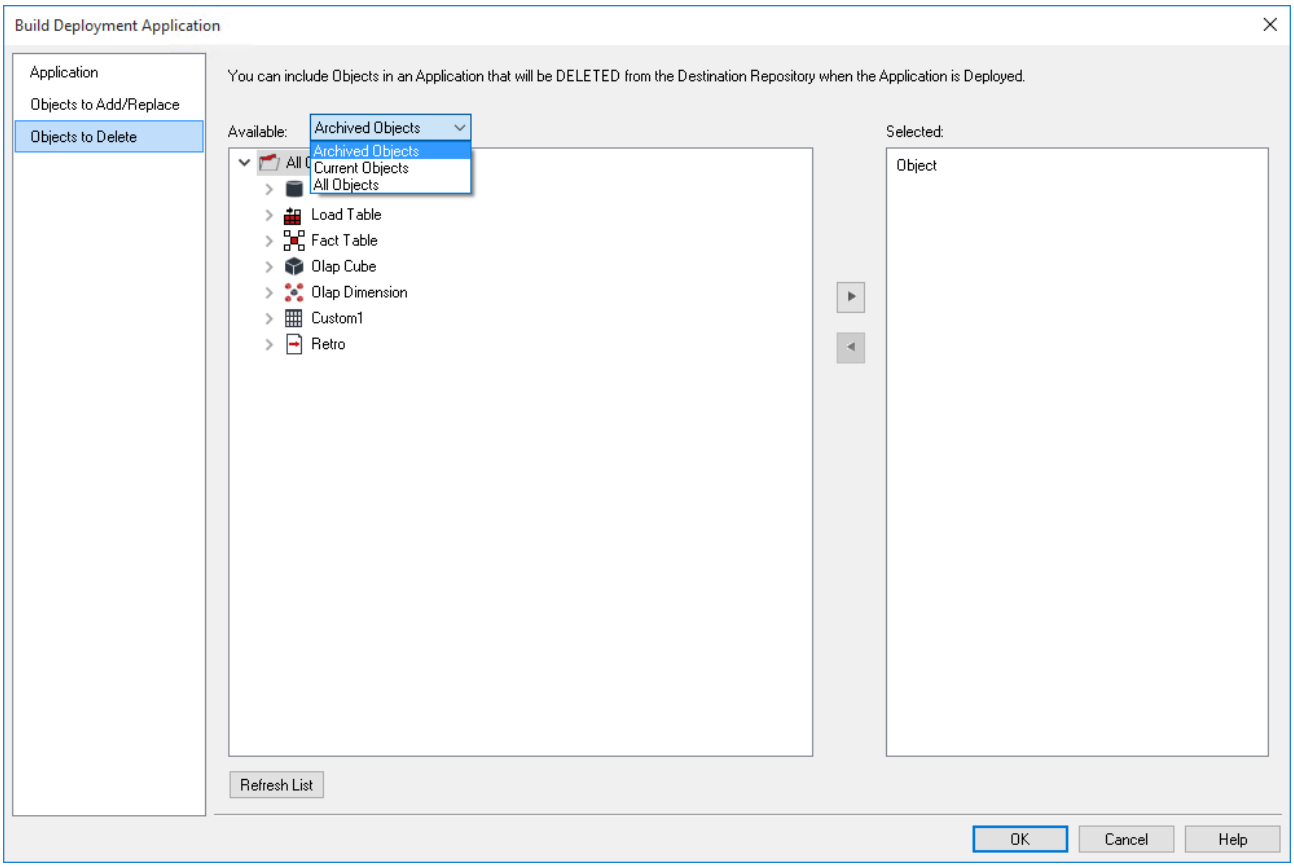
Notes

- Projects or Groups that have been flagged as inactive are not displayed in the **Available** pane. Refer to **Maintaining Group/Project Active Flag** for details.
- You can select multiple objects using Windows standard keyboard keys (CTRL or SHIFT) for multi selection.

© Copyright, WhereScape Software Ltd All rights reserved.

Objects to Delete

This tab enables you to select the objects to delete in the destination repository. Click the **Available** drop-down field to select an option for displaying objects in left object tree pane, each option is described in the table below. Double click a Project/Group folder to display the objects that are associated with them. Objects can be moved from the left object tree pane to the right pane by double-clicking an object or by using the **>** button.



Options	Description
Archived Objects	This option includes all objects that have been versioned and deleted. This is the case in a development repository, where the object was removed after versioning. This is the default option.
Current Objects	This option includes all the currently available objects and their associated Projects or Groups.
All Objects	This option includes all the **Current Objects** plus the **Archived Objects** (deleted / versioned). Objects that were versioned and deleted will appear in **All Objects** but will not appear under any Projects or Groups.

© Copyright, WhereScape Software Ltd All rights reserved.

Application Loading

Note

Applications can only be loaded into the same relational database type from which they where created. (For example, an Oracle application can only be loaded into an Oracle database).

Applications are loaded via the Setup Administrator utility. The normal process for implementing an application would be as follows:

1. Run the **Setup Administrator** utility.
2. Change the application directory to the application's location.
3. Turn ON logging in the Setup Administrator utility using **Tools > Start Logging**.
4. Load the application via the Setup Administrator utility.
5. Choose the level of metadata application. There are several levels, from load metadata only through to load metadata and apply changes to all tables.
6. Resolve any connections and tablespaces/filegroups to those present in the target environment.
7. Create/Re-create or Alter database tables, if selected in (5).
8. Compile database procedures, if selected in (5).
9. Turn OFF logging.
10. Review the output in the Setup Administrator utility.
11. Review the log file.

Note

Some database operations, such as converting an existing non-partitioned table to a partitioned table, cannot be done using a deployment application. In these cases, some manual intervention may be required to update the target databases to match the new metadata.

Refer to the **Setup Administrator** manual for more information about loading an application.

Creating and Loading Applications from the Command Line

It is possible to create and load applications from the command line by running a bat file. The WhereScape RED Application directory contains an example bat file WSL_Application_Create_Restore_Point_and_Load.bat for creating and loading.

Right-click this file and select **Edit** to see the steps outlined, as well as the details about the options available.

The first step **creates** a restore point application (R) based on objects about to be loaded. This process calls the command line functionality of RED and creates the application file.

The second step **loads** an application (A) into a test WhereScape RED repository. It uses an xml file to specify various options, calls the command line automation functionality of Setup Administrator and loads the application (A).

The WhereScape RED Application directory contains example xml files:

- WSL_Application_Load_SQL.xml for SQL Server
- WSL_Application_Load_TER.xml for Teradata
- WSL_Application_Load_ORA.xml for Oracle

The tags in the xml file must be edited as the login/connection details, etc. must be correctly defined.

The Batch Application Create options and the Batch Application Load options are listed at the end of the batch file. The values for these variables must also be customized before running the file.

© Copyright, WhereScape Software Ltd All rights reserved.

The process typically involves the following steps:

1. Create an application (A) in RED containing your data warehouse changes.
2. RUN WSL_Application_Create_Restore_Point_and_Load.bat from the command line; which creates a restore point application (R) and applies application (A) to a test WhereScape RED repository.
3. If the changes are incorrect, they can be undone by loading the restore point application (R).

Batch Application CREATE Options

The following options are available in RED to create applications.

Option	Description
/BA	Selects batch application create.
/U	ODBC user name.
/P	ODBC password.
/C	ODBC DSN name.
/A	DSN architecture
/M	Metadata database for Teradata logon.
/N	RED User name.
/D	Directory to save application files.
/I	New application files identifier.
/V	New application files version.
/AP	Project name - all objects in a Project; and associated jobs.
/AG	Group name - all objects in a Group; and associated jobs.
/ALL	All objects - all objects, jobs and parameters.
/RC	Remove connections - drop all connections from the application.
/RJ	Remove jobs - drop all jobs from the application.
/RP	Remove parameters - drop all parameters from the application.
/AF	Absolute application id file name which restore point is being created for.

If using a trusted connection, the user and password are not necessary.

Batch Application LOAD Options

The following options are available in Setup Administrator to load applications.

Option	Description
/AL	Select application load.
/AF	Absolute application ID file name.
/LF	Absolute log file name.
/PF	Absolute xml parameter file name.
/SO	Stay open - Setup Administrator does not close after the application has been created. Relevant for /AL LOAD option. Relevant for -QA, -QR, -WS, -QL and -QV application switch options. Not relevant for -QS switch option. For switch options, refer to **SQL Server Quick Application** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

Setup Administrator Process Exit Codes

When installing an application using the command line, the Setup Administrator process returns one value from a set of exit codes from its process when exiting.

After an application install, when the Setup Administrator process exits, users can check the exit codes to determine the cause for a possible problem with the application install.

The set of Setup Administrator exit codes are described below:

Exit Code Name	Exit Code	Description
AdmExitedNormally	0	The user clicked the main window's X button in the top right, the Exit item from the File menu, or the task that ran completed without error.
InternalError	1	An unexpected internal error occurred. The user should contact WhereScape support supplying any relevant log files.
CommandCancelled	2	The user clicked the cancel button from one of the presented dialog boxes, thus aborting the running task.
RequiredParameterMissing	3	One or more required parameters is missing e.g. a name for the Scheduler, an ODBC DSN, a user name, a password.
OdbcError	4	An error was reported from the RDBMS that Setup Administrator was communicating with.
FileIoError	5	A specified file or directory was unable to be opened, read from or written to.
ServiceControlManagerError	6	An error was reported from the Windows Service Control Manager. Often, this is because Setup Administrator needs to be run "As Admin" when dealing with the SCM.
OracleCompilationError	7	An Oracle stored procedure has failed to compile.
MetaDataError	8	Metadata was/wasn't found in the specified RDBMS, or the metadata failed to upgrade, or the metadata is corrupt, or the metadata type mismatches i.e. trying to import an Oracle application into Teradata metadata.
ItemsCheckedOut	9	In batch mode, if you want to overwrite checked out items, you must explicitly state so by setting the option to true in the settings.
NotLicensed	10	License does not permit objects of a specific type to be imported.

Users can test for these values using a batch file and take action accordingly.

Below is an example batch file that can be used to check Setup Administrator's exit code.

Example batch file - SQL:

```
@echo off


Adm.exe /AL /AF "app_con_SQLT_base_1.wst" /LF "ImportLog.txt" /PF
"WSL_Application_Load_SQL.xml"


if errorlevel 8 goto MetaDataError
if errorlevel 5 goto FileIoError
if not errorlevel 0 goto ImportError


echo Successfully imported
goto end


:MetaDataError
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
sendmail -recipient admin@company.com -subject "Import failed" -body "Corrupt meta data
found during application import. Re-install the Data Warehouse"
goto end


:FileIoError
sendmail -recipient admin@company.com -subject "Import failed" -body "ExportedDiagram.xml
is missing, re-generate it then re-run this process"
goto end


:ImportError
sendmail -recipient admin@company.com -subject "Import failed" -body "Unable to import the
application file ExportedDiagram.xml"
goto end


:end
```

Example batch file - 3D:

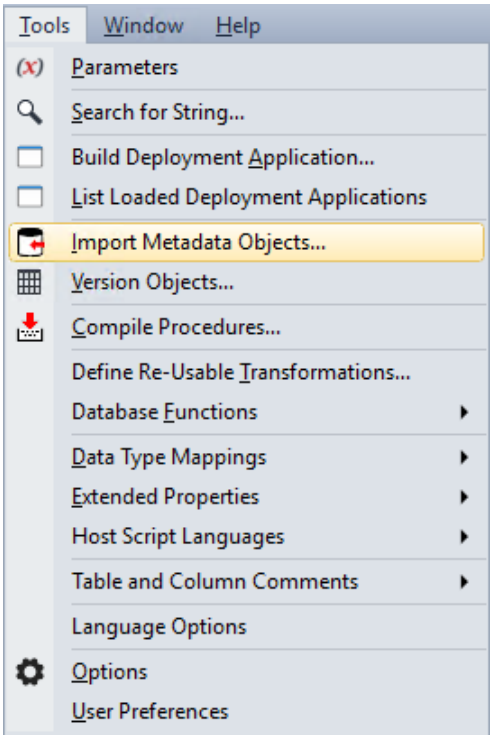
```
@echo off
Start /wait /D "C:\Program Files (x86)\WhereScape\Adm.exe" /AL /AF "ExportedDiagram.xml"
/LF "C:\temp\test\ImportLog.txt"
if %errorlevel% equ 8 goto MetaDataError
if %errorlevel% equ 5 goto FileIoError
if not %errorlevel% equ 0 goto ImportError
echo Successfully imported
goto end


:MetaDataError
sendmail -recipient admin@company.com -subject "Import failed" -body
"Corrupt meta data found during application import. Re-install the Data Warehouse"
goto end


:FileIoError
sendmail -recipient admin@company.com -subject "Import failed" -body
"ExportedDiagram.xml is missing, re-generate it then re-run this process"
goto end


:ImportError
sendmail -recipient admin@company.com -subject "Import failed" -body
"Unable to import the application file ExportedDiagram.xml"
goto end


:end
```

© Copyright, WhereScape Software Ltd All rights reserved.

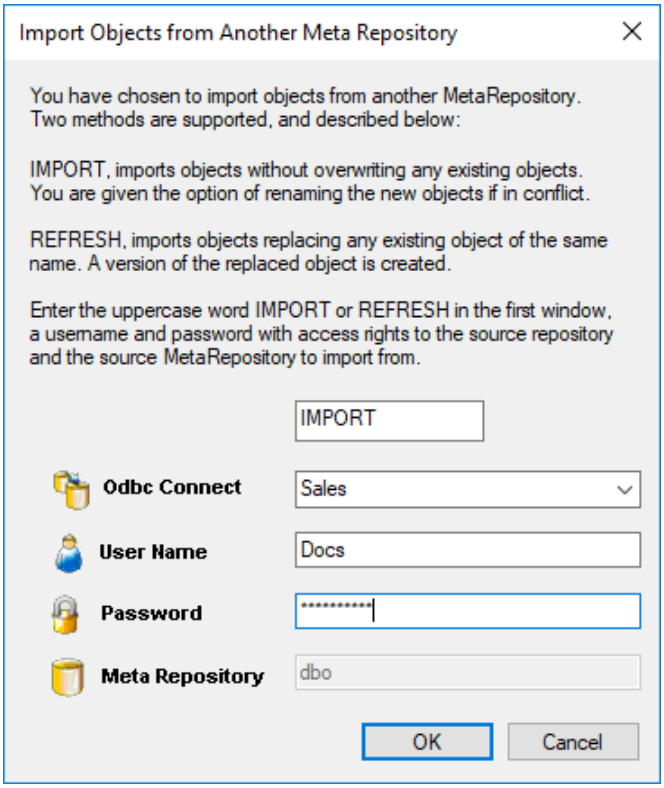
Importing Object Metadata

Any group of objects can be imported into the current metadata repository from another repository. If an object already exists in the target repository, then it is either skipped or replaced depending on the type of import undertaken. If an object is to be replaced as part of an import, a version of the object is created prior to its replacement.

To import an object or group of objects select the **Tools > Import Metadata Objects** menu option.



A window appears which provides two options, **IMPORT** or **REFRESH**. An import will not replace an existing object of the same name. A refresh will version and replace any existing object of the same name.



© Copyright, WhereScape Software Ltd All rights reserved.

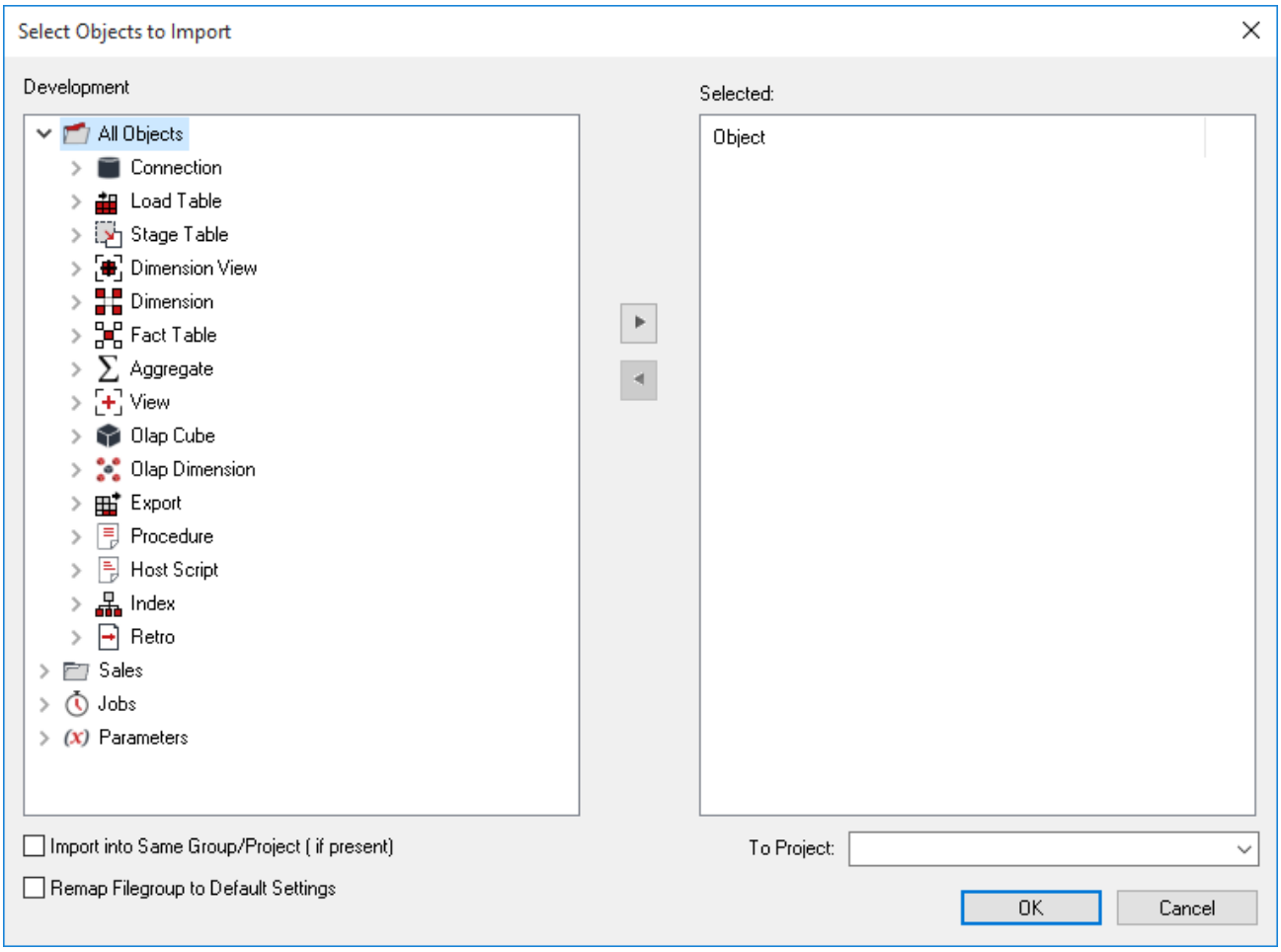
Enter the connection, and a database user name and password that has access to the source metadata repository.

Finally, enter the user name of the metadata repository you want to import from. In most situations, the **User Name** and **Meta Repository** would be the same.

However, if you only have read access to a meta repository, then it may be necessary to log in to the database under a different user name from that of the repository you are trying to import from.

You are not permitted to select the current meta repository in the **Meta Repository** field, but are permitted to log in using the existing repository username. Once a successful log on is completed, the contents of the source repository are loaded and the following window appears.

Select an object by double-clicking on it or by selecting and using the **>** button. If an object such as a table is chosen, then any related scripts, procedures and indexes are also selected.
They can be removed if required. A target project can be selected.



Once all required objects are selected, the import will commence when the **OK** button is clicked.

On completion, a window appears which displays the number of each type of object imported, and skipped.

Note

The repository from which you are importing must be the same metadata version as the target repository.

© Copyright, WhereScape Software Ltd All rights reserved.

Importing Language Files

Note

Applications can only be loaded into the same relational database type from which they were created. (For example, an Oracle application can only be loaded into an Oracle database).

Language Files are loaded via the **Setup Administrator** utility. The normal process for implementing a Language file would be as follows:

1. Run the **Setup Administrator** utility.
2. Go to the **Languages** menu item in the top command bar and select **Load Languages**.
3. Right-click the Language file to be loaded and select **Install Language**.
4. Select the **ODBC** data source and Log on to the target meta repository.
5. Select the language to be updated.
6. Review the output in the **Setup Administrator** utility.

Refer to the RED Installation Guide for more information about loading a Language File.

Data Warehouse Testing

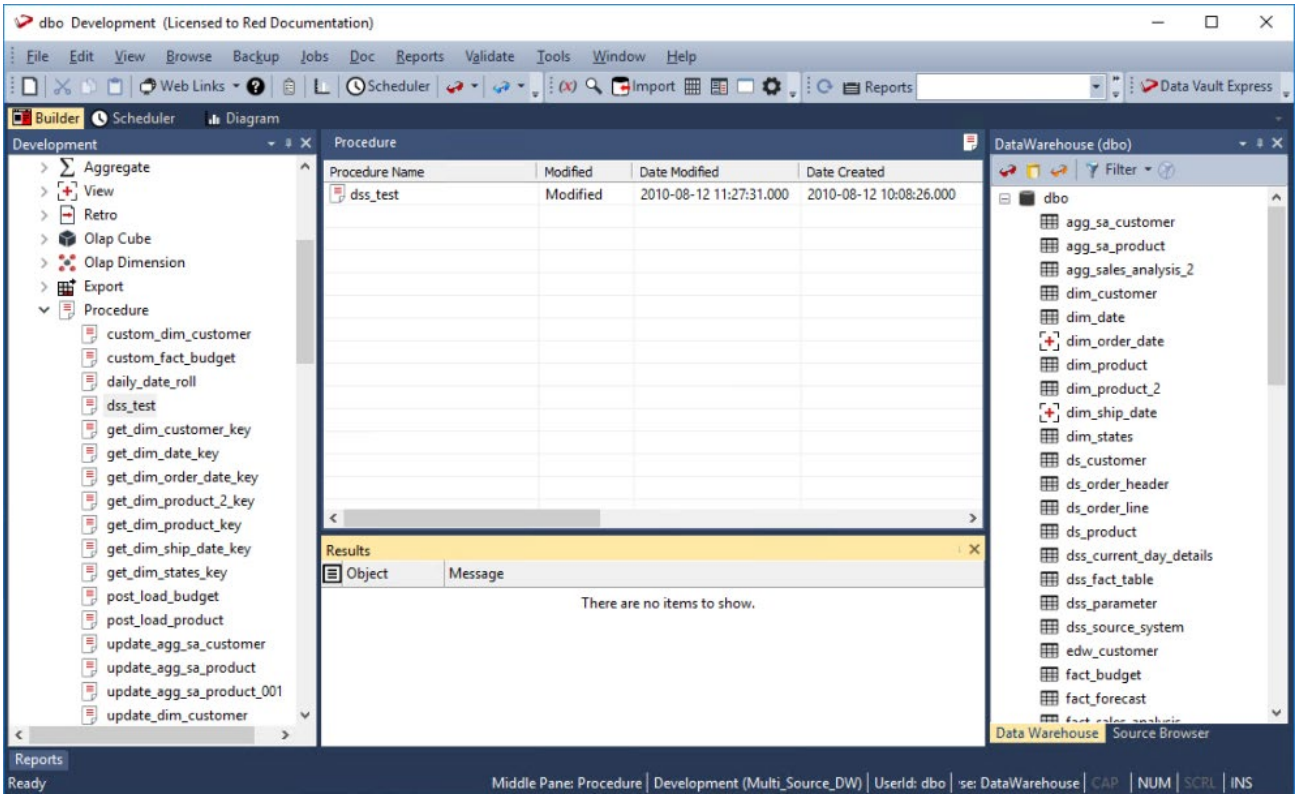
Note

Applications can only be loaded into the same relational database type from which they where created. (For example, an Oracle application can only be loaded into an Oracle database).

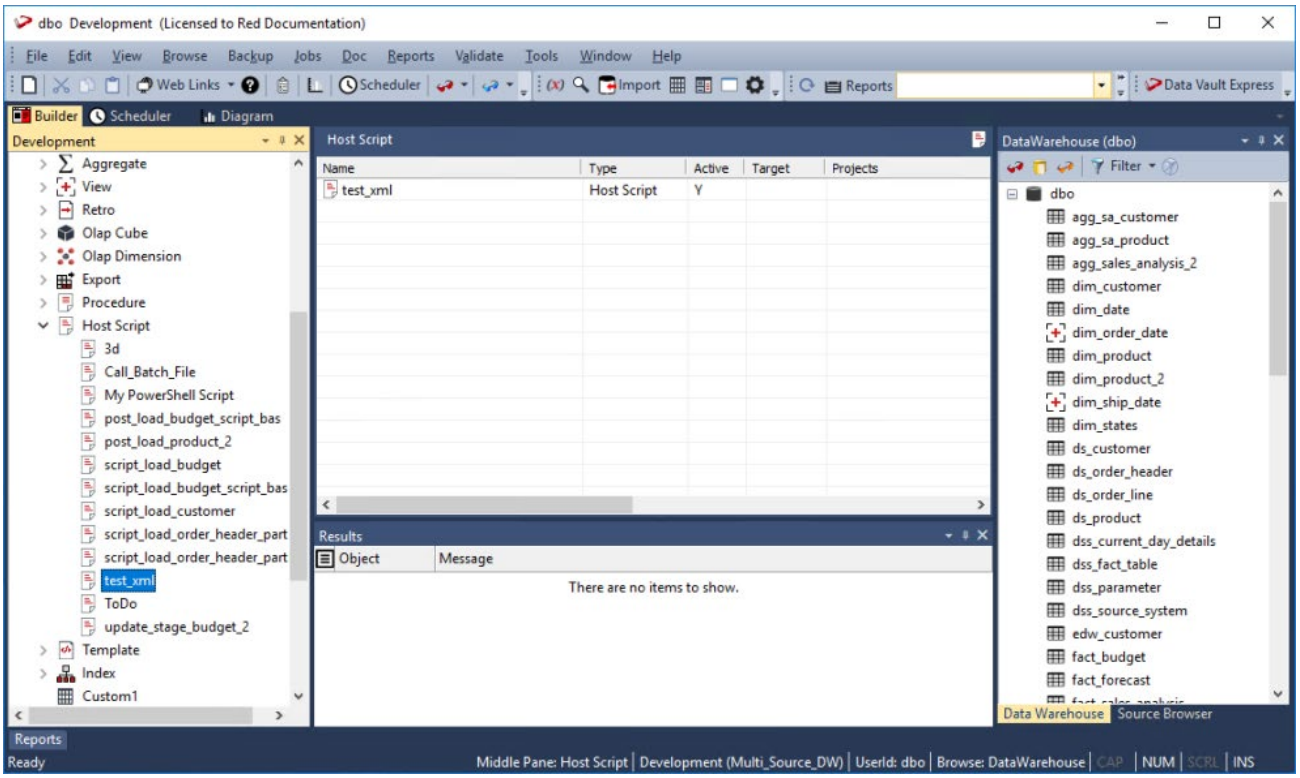
Testing applications are loaded via the **Setup Administrator** utility. Refer to the RED Installation guide for more information on how to load an application.

A testing application set consists of a Procedure and an XML script and provides the ability to define a series of tests against data warehouse objects; either comparing them to an expected value or to the results of a query.

Once the application set has been loaded, the Procedure and the XML script will be visible in the left pane.



© Copyright, WhereScape Software Ltd All rights reserved.

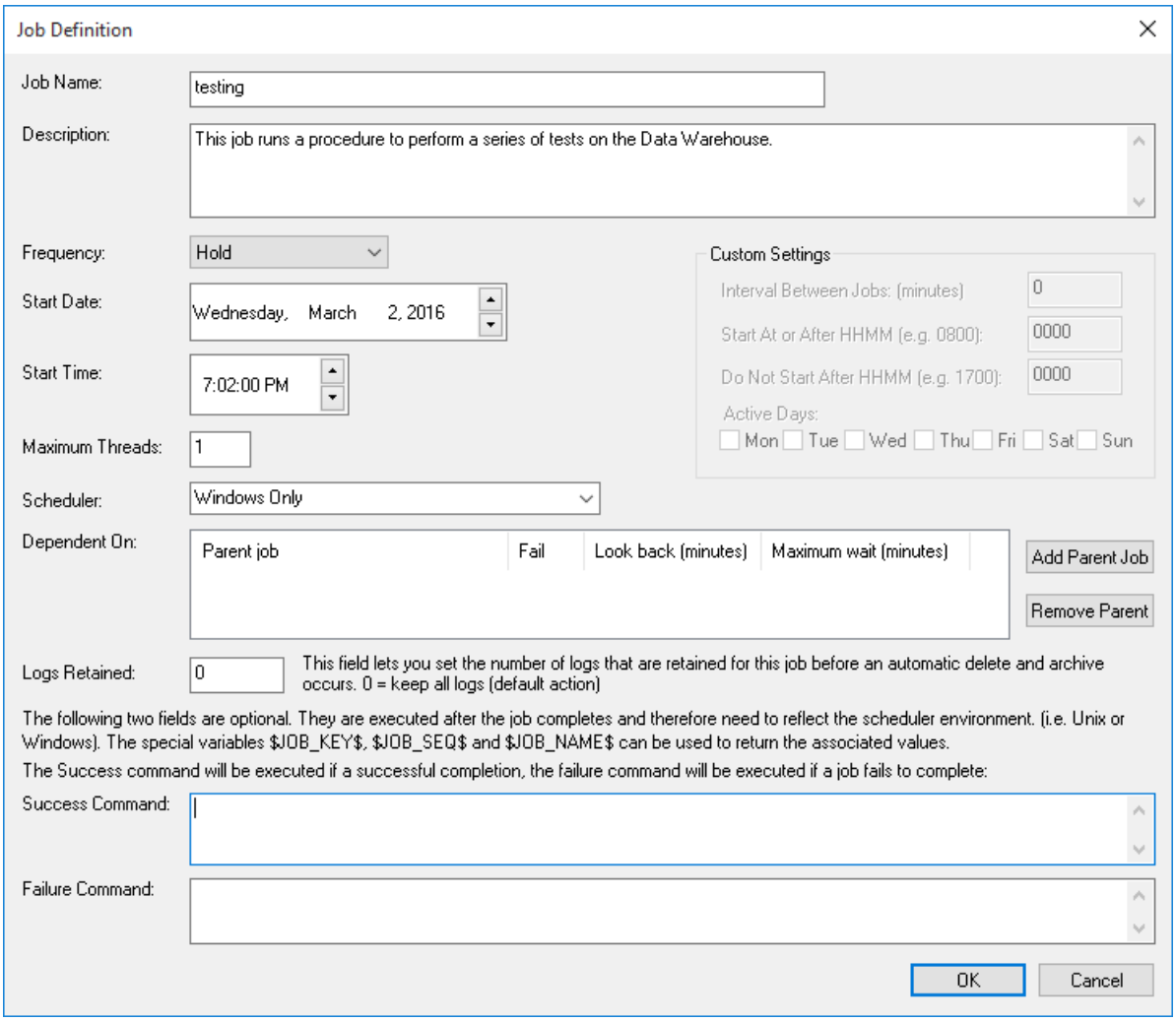


The XML script contains the test definitions. Each test is a new XML node in the comparison query. The procedure simply runs the test and determines whether the tests are passed or not. This is most likely to be run as a scheduled job within WhereScape RED.

To create a job:

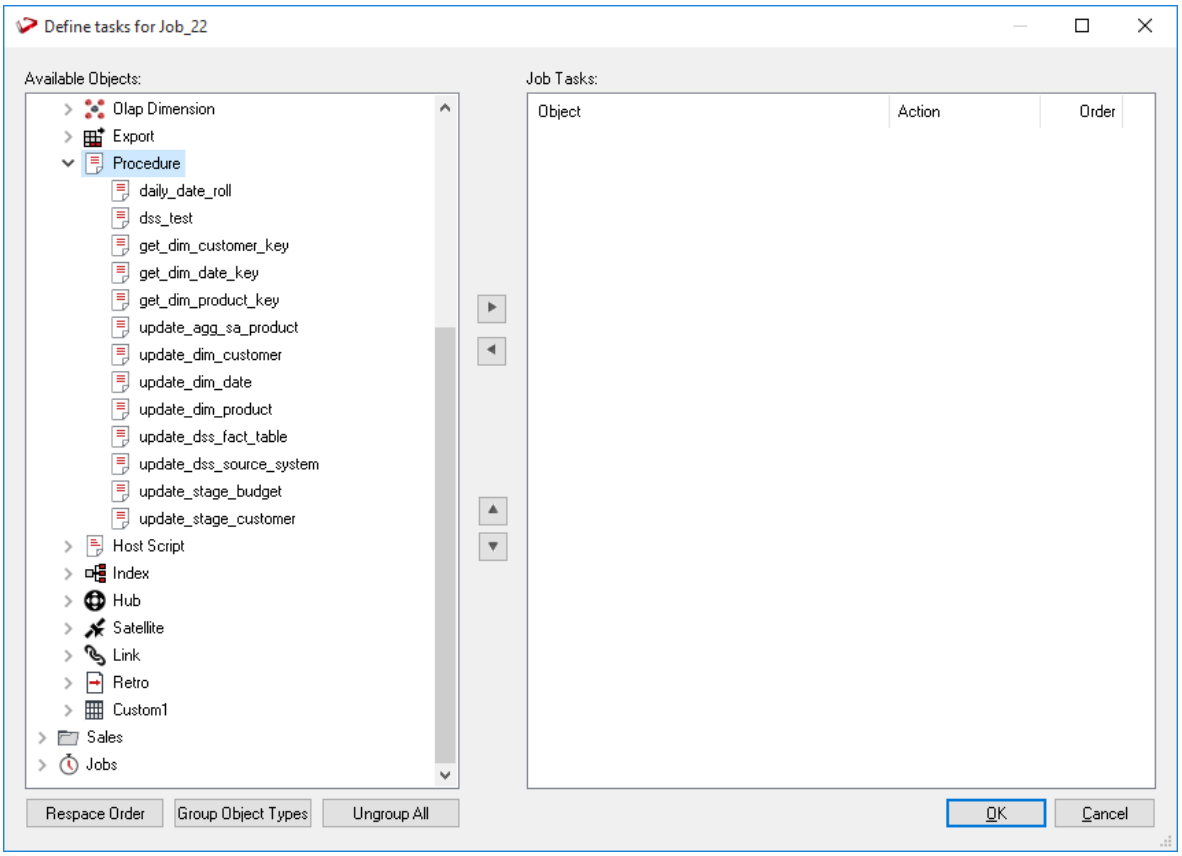
1. Click the **Scheduler** button.
2. Choose **File** and then **New Job**.
3. Enter the definition of the job.

© Copyright, WhereScape Software Ltd All rights reserved.



4. To select the test procedure as a task, open the Procedure object heading in the left pane. Choose **dss_test** and the **>** button. Click **OK**.

© Copyright, WhereScape Software Ltd All rights reserved.



5. To run the job, click on the **All Jobs** button and then right-click on the job and select **Start the Job**.

© Copyright, WhereScape Software Ltd All rights reserved.

Backing Up and Restoring Metadata

This chapter covers the saving and reloading of metadata repository objects. The backup section describes the methods for backing up the metadata repository. It can also be backed up via normal database backup procedures. The restore section covers the metadata restoration functions available.

WhereScape RED provides two different methods for backing up and restoring metadata:

1. Backup and Restore using the native backup tools available in the data warehouse database.
2. WhereScape RED unload and load.

Backup using DB Routines

The backup of the metadata repository can be undertaken as a separate exercise from the general backup of the data warehouse. The metadata backup does not backup any of the actual data warehouse tables. It only saves the table definitions, index definitions, procedures etc., so is not normally large.

The backup includes any tables that begin with "dss_". In this way, key metadata tables such as dss_parameter, dss_source_system and dss_fact_table are included in the backup. These tables are required if the restored metadata is to function correctly.

It is recommended that the metadata is backed up at least daily when possible. There are two methods for backing up the meta repository. These are through a scheduled UNIX task, and through the main WhereScape RED tool.

UNIX Backups

Where the data warehouse database is running on a UNIX server, it is possible to schedule a regular backup of the metadata. The UNIX scripts shipped with WhereScape RED include a script called from cron to perform daily backup of the metadata. These backups are in turn compressed and saved for 60 days by default. Refer to the RED Installation Guide for details on these scripts and their setup.

Windows Backups

Two main methods of Windows backup exist within WhereScape RED. The first is a database independent backup, which is designed purely for moving the meta repository, and should not be used for regular backups. The other method is the database specific backup. See the following sections for the appropriate database.

Oracle Windows Backups

An Oracle export based backup can be taken from the WhereScape RED tool by selecting the **Backup > Export the metadata (Oracle Export)** menu option. This option assumes that the Oracle Client on the PC is the same version as that of the Oracle database where the WhereScape RED meta repository is stored. The backup may not work if the versions differ.

When executed, this menu option attempts to locate the Oracle export utility. This utility is normally called 'exp' and resides in the Oracle bin directory. If WhereScape RED cannot locate this utility or it has not been loaded onto the PC, then the export does not proceed, and an error message is displayed.

A pop-up window asks for a file name for the export. A directory should be chosen, and a name entered. When the **Save** button is clicked, the export will start. The following files are created (where 'file' is the name of the chosen file). A dialog box appears to show the results of the backup.

File name	Purpose
file.bat	Windows command file containing the export command.
file.dbl	Command file for the re-creation of any database links.
file.drp	Command file for dropping all metadata tables. (Used in the restore prior to a reload).
file.exp	Oracle export file.
file.log	Log file of the export.
file.par	Parameter file for the export. Contains an entry for each meta table to be exported.

© Copyright, WhereScape Software Ltd All rights reserved.

File name	Purpose
file.pro	A file containing each procedure stored in the metadata.
file.seq	Command file to re-create the sequences used by both the metadata and the data warehouse tables.

If problems are encountered with the backup, it may be possible to manually run the generated .bat script file to ascertain what has gone wrong.

SQL Server Windows Backups

A SQL Server bcp based backup can be taken from the WhereScape RED tool by selecting the **Backup > Export the metadata (Sql server bcp)** menu option. This option assumes that the SQL Server bcp utility is available on the PC.

When executed, this menu option attempts to locate the SQL Server bcp utility. This utility is normally called 'bcp' and resides in the SQL Server bin directory. If WhereScape RED cannot locate this utility or it has not been loaded onto the PC, then the export does not proceed, and an error message is displayed.

A pop-up window asks for a directory name for the export—a directory must be chosen or created. The export starts when the **Save** button is clicked. The following files are created. A window appears to show the results of the backup.

File name	Purpose
wsl_bcp_out.bat	Windows batch file used to run all the bcp table exports
wsl_bcp_out.err	Errors from the bcp export of each table
wsl_bcp_out.log	Log file of the bcp table exports
***.bcp**	Each metadata table is stored with a .bcp extension. This is the export of each table.
wsl_bcp.par	Parameter file for the export. Contains an entry for each meta table to be exported. Unused by SQL Server.
wsl_bcp.drp	Command file for dropping all metadata tables. (Used in the restore prior to a reload).
wsl_bcp.pro	A file containing each procedure stored in the metadata.
wsl_bcp.seq	Unused by SQL Server.
wsl_bcp.dbl	Command file for the rebuild of all the linked servers. Used during the restore.
wsl_bcp.bld	Basis of the command file that will be used to restore the metadata. This file is modified to include the target directory prior to restore.

If problems are encountered with the backup, it may be possible to manually run the generated .bat script file to ascertain what has gone wrong.

DB2 Windows Backups

A DB2 IMPORT based backup can be taken from the WhereScape RED tool by selecting the **Backup > Export the metadata (DB2 export)** menu option. This option assumes that the IBM DB2 EXPORT command is available on the PC.

When executed, this menu option attempts to locate the SQL Server bcp utility. This utility is normally called **bcp** and resides in the SQL Server bin directory. If WhereScape RED cannot locate this utility or it has not been loaded onto the PC, then the export does not proceed and an error message is displayed.

A pop-up window asks for a directory name for the export. A directory should be chosen or created. When the **Save** button is clicked, the export will start. The following files are created. A dialog box appears to show the results of the backup.

File name	Purpose
RED.bat	Windows batch file used to run the DB2 EXPORT for each metadata table.
RED.cmd	Control file for the export. Contains an entry for each meta table to be exported.
RED.log	Log file of the EXPORT.
RED.pro	A file containing each procedure stored in the metadata.

© Copyright, WhereScape Software Ltd All rights reserved.

File name	Purpose
***.ixf**	Each metadata table is stored in a file with a ixf extension. This is the export of each table.
restore.drp	Command file for dropping all metadata tables. (Used in the restore prior to a reload).
restore.lst	Basis of the command file that will be used to restore the metadata. This file is modified to include the target directory prior to restore.

If problems are encountered with the backup, it may be possible to manually run the generated RED.bat script file to ascertain what has gone wrong.

Restoring DB Backups

WhereScape RED metadata can be restored from a prior backup. For Oracle data warehouses, the restore can take place either from UNIX or through the WhereScape RED tool. The same set of files as defined in the backup section are used for both restores and can be moved between the Windows and UNIX environments. For SQL Server data warehouses, the restore must take place in a Windows environment either through the WhereScape RED tool or manually.

Note: If transferring the export files via ftp, ensure that the file.exp and .bcp files are transferred in binary mode and that the other files are transferred in ASCII mode.

Restoring metadata

A backup of a metadata repository can be restored over the top of an existing repository. This action replaces the existing repository in its entirety. The restore of repositories is covered in an earlier section, but its uses in the promoting of code are discussed here. This restore process does not affect any existing database tables or any compiled procedures, so it can be used as a means of updating a metadata repository, albeit by replacing that repository. It is often a good method to choose when first establishing a new repository. For example, if we have a development repository and want to create a production environment the steps may be:

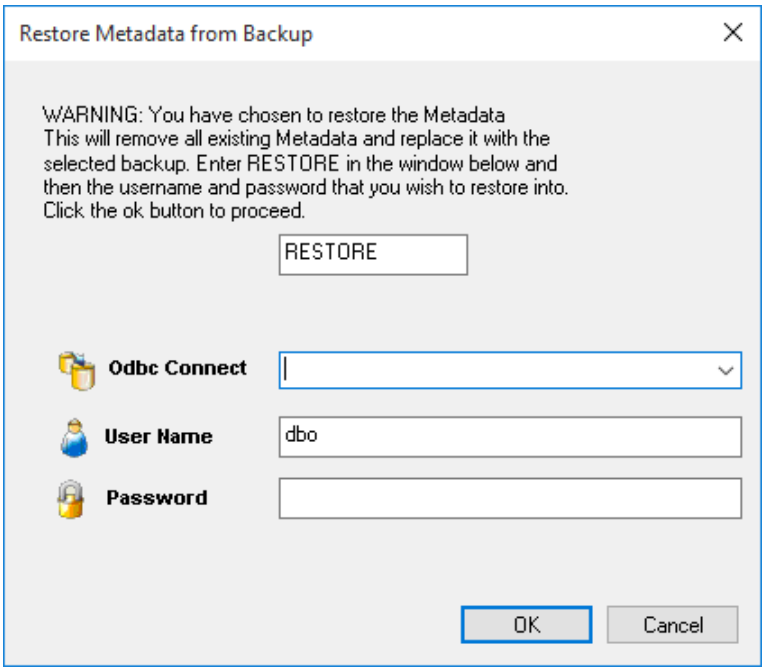
1. Backup the development repository, using either the UNIX or Windows repository backup. Refer to the previous sections.
2. Create a new repository using the **Setup/Administrator** utility.
3. Log on to the newly created repository.
4. Restore the backup of the development repository into the new repository. For Oracle this requires the database privilege 'imp_full_database', or the dba role.
5. Modify any connections, set any default values, alter any table sizes or extents.
6. Create the tables.
7. Compile all procedures.

Oracle Windows Restore

Select the menu option **Backup/Restore Metadata (from Oracle Export)** to begin the restore process.

A dialog box will appear. The word **RESTORE** needs to be entered along with the **User Name** and **Password** where the metadata is to be restored to. The **User Name** does not have to be that of the current metadata repository, but it must be a valid repository. If a restore is being performed to a repository that differs from that of the backup then the 'imp_full_database' database privilege may be required.

© Copyright, WhereScape Software Ltd All rights reserved.



Once the **OK** button is clicked, a new dialog box will appear asking for a selection of the export file. Browse to the directory where the export is located and select the export file. Once the export file is selected, the import will begin. A dialog box will appear to show the results of the import.

Oracle UNIX restore

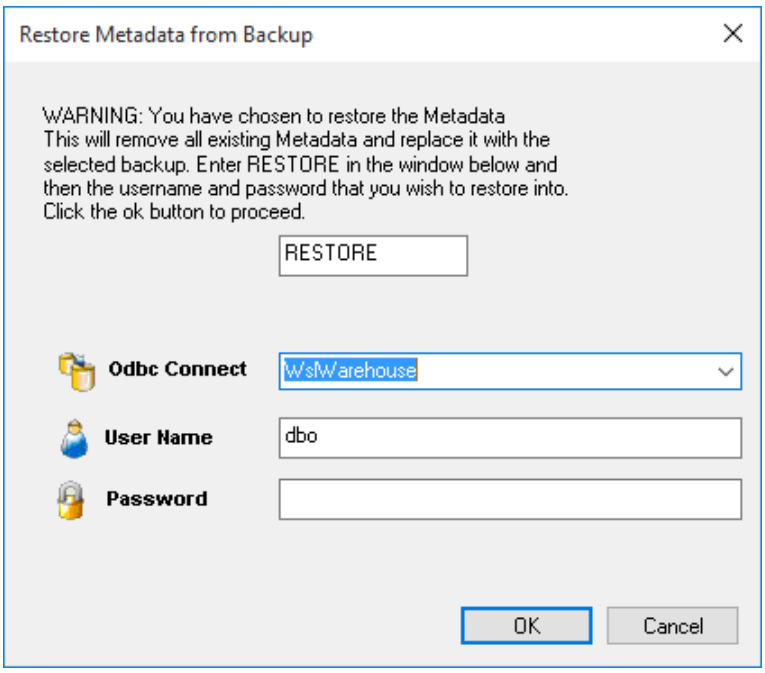
The UNIX restore is actioned via the meta_restore_010.sh shell script. This script expects the export file to be located in the wsl/export directory, so it may need to be un compressed and moved to this directory if not already present.

Execute the meta_restore_010.sh shell script. You will be prompted to enter the backup file name. Once successfully entered a username and password to be restored into are requested and the restore commences.

SQL Server Windows Restore

Select the menu option **Backup/Restore the metadata (from Sql server bcp)** to begin the restore process.

A dialog box will appear. The word **RESTORE** needs to be entered along with the connection, **User Name** and **Password** where the metadata is to be restored to. The metadata will always be restored to the dbo schema. The connection does not have to be that of the current metadata repository, but it must be a valid repository.



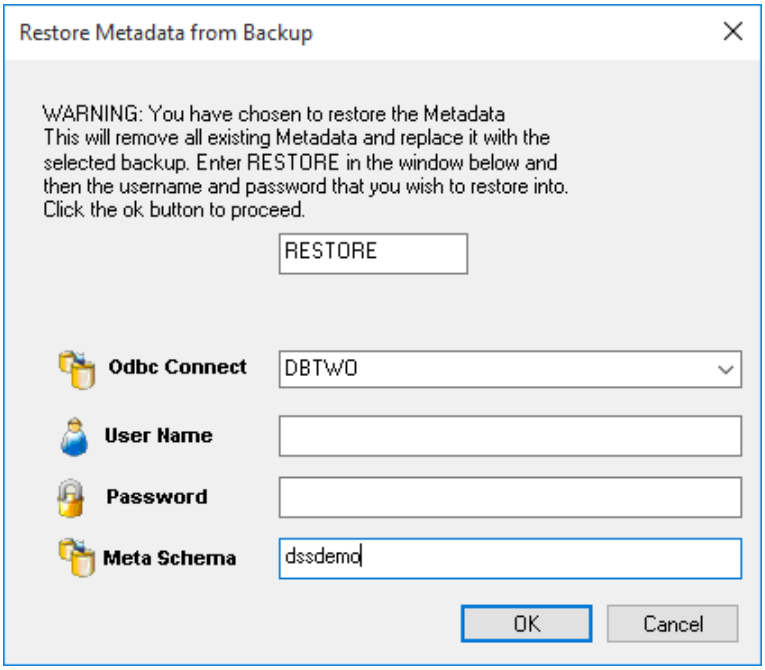
© Copyright, WhereScape Software Ltd All rights reserved.

Once the **OK** button is clicked a new dialog box will appear asking for a selection of the export directory. Browse to the contents of the directory where the export is located. Once the export directory is selected, the import will begin. A dialog box will appear to show the results of the import.

DB2 Windows Restore

Select the menu option **Backup/Restore metadata (from DB2 export)** to begin the restore process.

A dialog box will appear. The word **RESTORE** needs to be entered with the **User Name** and **Password** for DB2. If using a trusted connection to DB2, the username and password should be left blank. The metadata schema should also be entered. The schema does not have to be that of the current metadata repository, but it must contain a valid repository.



Once the **OK** button is clicked, a new dialog box will appear asking for a selection of the export file. Browse to the directory where the export is located and select the export file. Once the export file is selected, the import will begin. A dialog box will appear to show the results of the import.

Unloading Metadata

WhereScape RED provides a generic unload utility for backing up the metadata. The advantage of this backup is that it is database, and database version independent. It can be used to backup the metadata regardless of the version of the database client running on the PC. It is possible to transport the metadata from one database platform to another using unload and load. For example, the metadata from a SQL Server unload can be loaded into an Oracle database.

After performing a cross platform unload and load:

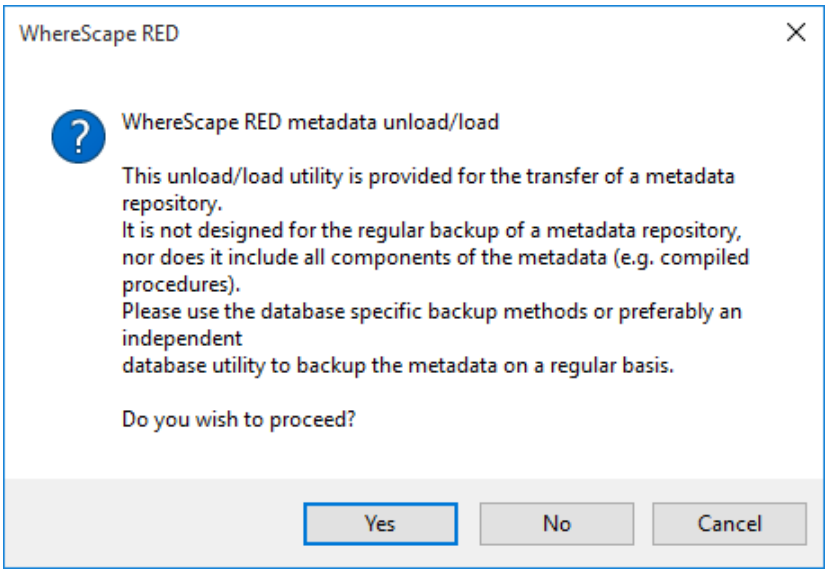
- Transformations must be altered manually to use the correct syntax for the new database.
- Generated procedures regenerated.
- Modified and custom procedures must be changed to use the procedure language of the new database platform.

© Copyright, WhereScape Software Ltd All rights reserved.

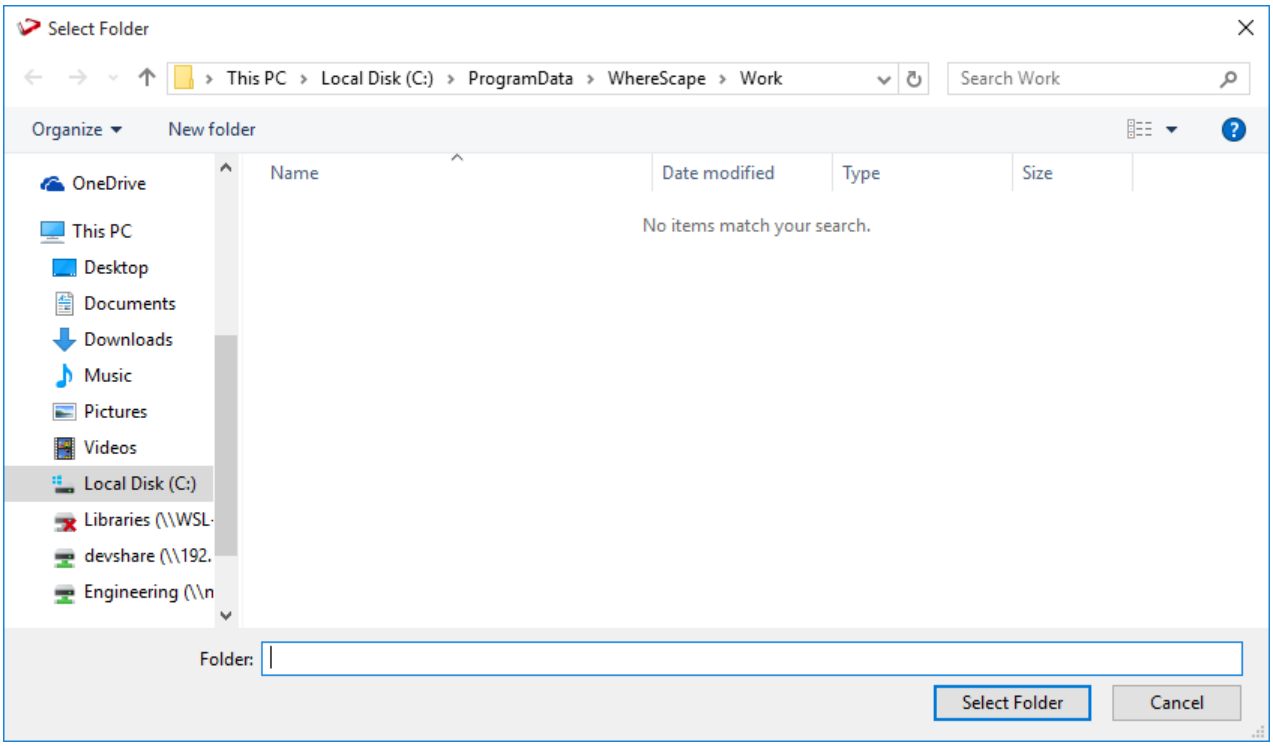
Performing an Unload

An unload can be performed within the WhereScape RED tool by selecting the **Backup > Unload the Metadata to Disk** menu option.

A popup window asks for confirmation to proceed with the unload. Click **Yes**.

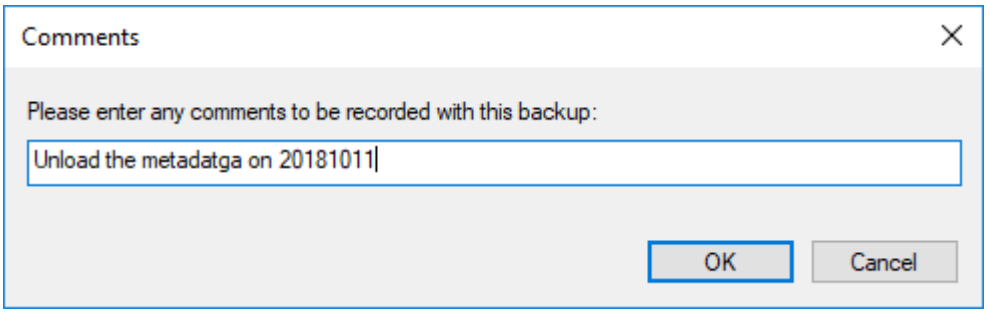


An additional pop-up window asks for the folder where the metadata will be exported. Select a directory and then click **Save**.

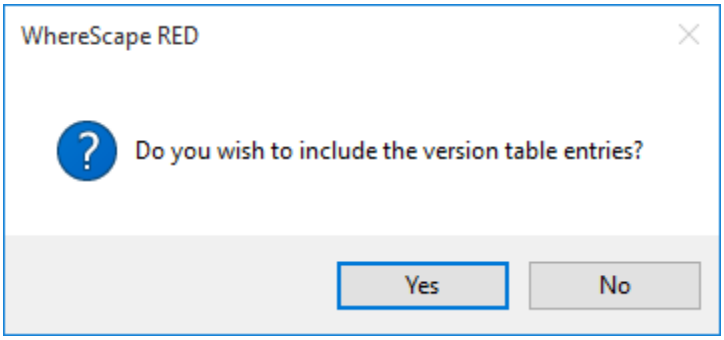


© Copyright, WhereScape Software Ltd All rights reserved.

Enter a comment for the unload and click **OK**.



Finally, click **Yes** or No on the include version history dialog:



This either includes or excludes version history metadata in the unload.

The unload starts and a progress bar is displayed to show the status.

© Copyright, WhereScape Software Ltd All rights reserved.

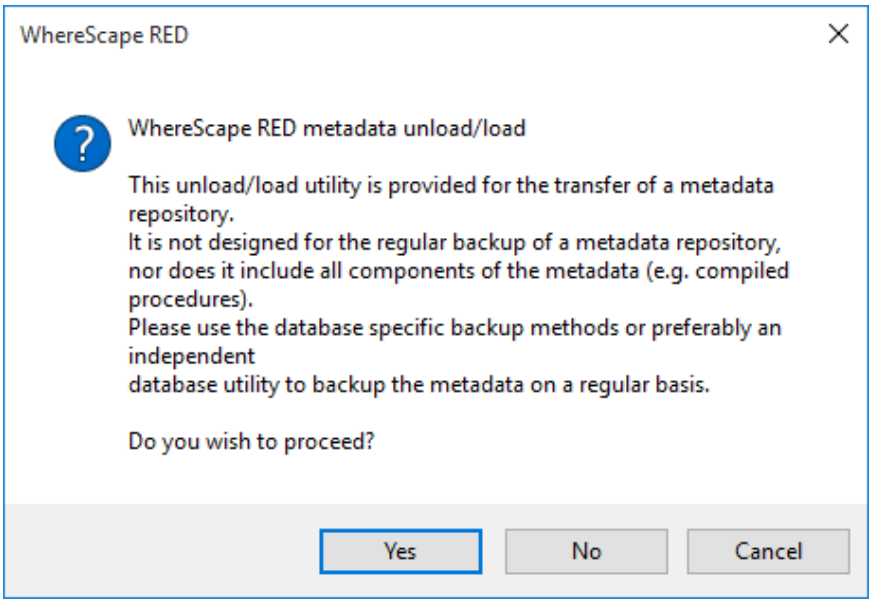
Loading an Unload

WhereScape RED metadata can be restored from a prior unload.

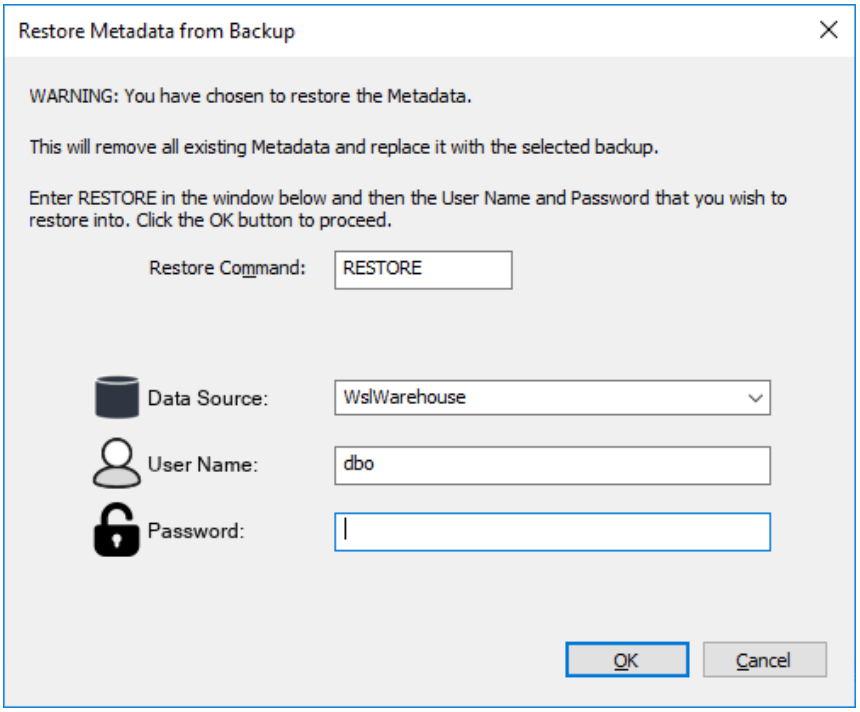
Performing a Load

An unload of a metadata repository can be loaded over the top of an existing repository. This action replaces the existing repository in its entirety. To load an unload, select the menu option **Backup > Load the Metadata from disk** to begin the load process.

A popup window asks for confirmation that a load is intended. Click **Yes**.



A dialog box appears. The word **RESTORE** needs to be entered. The ODBC connection needs to be chosen, along with the **User Name** and **Password** where the metadata is to be restored to. The username does not have to be that of the current metadata repository, but it must be a valid repository. Click **OK**.

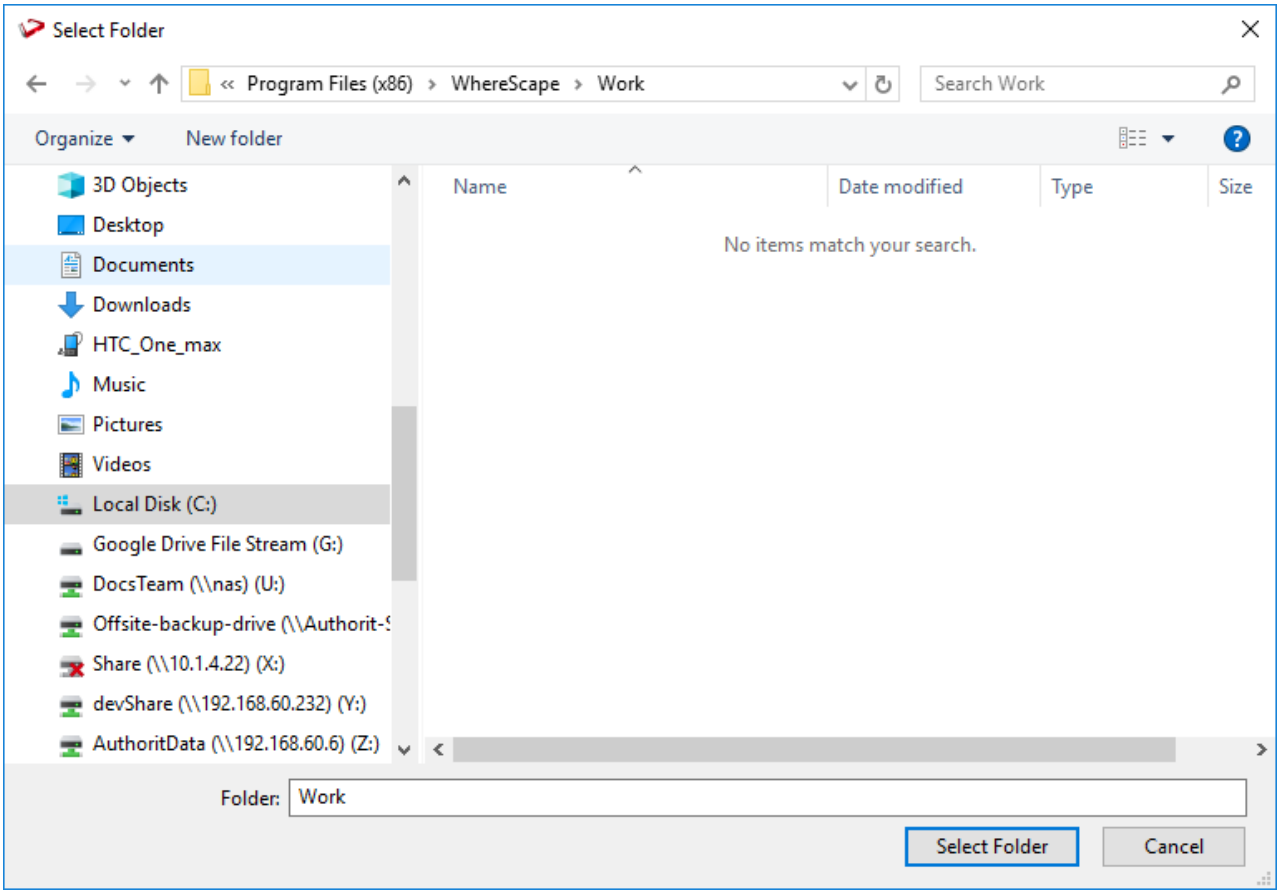


Note:

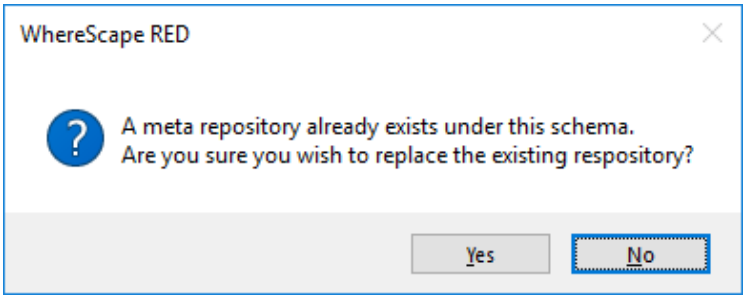
The dialog box above will include an additional field for **Schema** if loading into DB2.

© Copyright, WhereScape Software Ltd All rights reserved.

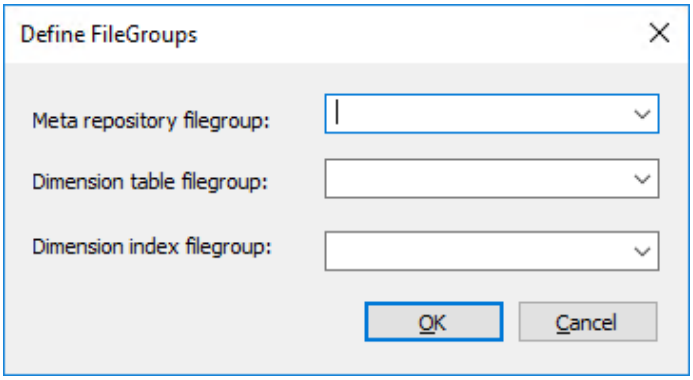
The next dialog box asks for the folder for the metadata to be loaded from. Browse to the contents of the directory where the unload is located and click **Select Folder**.



Confirm the load will overwrite the existing repository by clicking **Yes**:



Confirm the metadata and dimension tablespaces/filegroups for the metadata load and click **OK**:

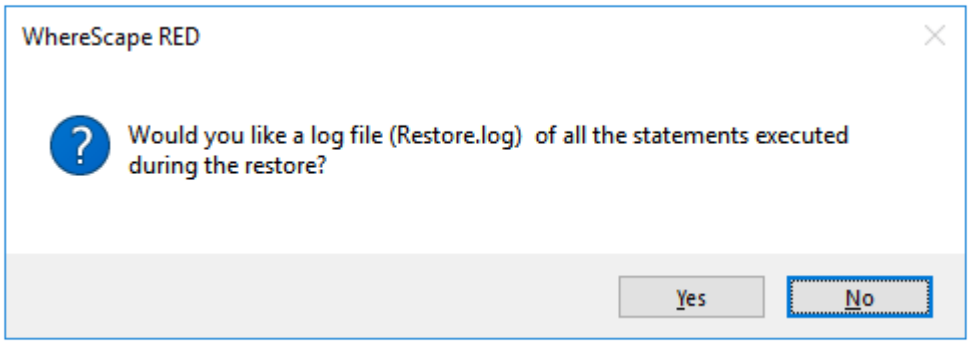


Note

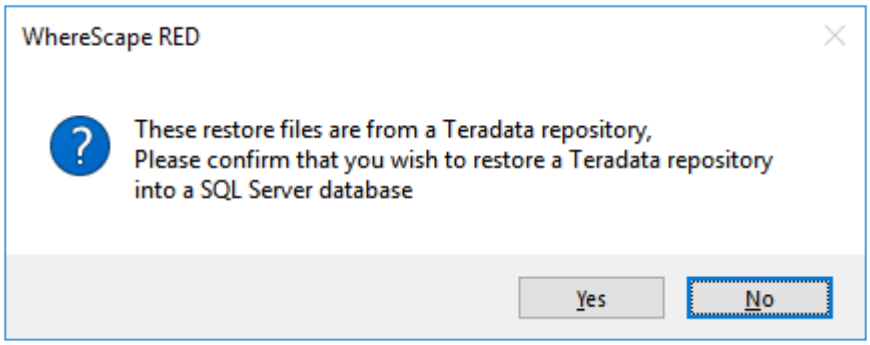
The dialog above is displayed when loading into SQL Server. It may differ slightly, if loading into Oracle or DB2.

© Copyright, WhereScape Software Ltd All rights reserved.

A pop-up window asks if a restore log is required. Click **Yes** for a log, or No otherwise.



If a cross platform load is being performed, the following dialog is displayed. Click **Yes**.



The metadata load now runs. Once the load has completed, start WhereScape Administrator and validate the metadata repository that has just been loaded.

© Copyright, WhereScape Software Ltd All rights reserved.

Altering Metadata

This chapter provides information on how to change and manipulate the data warehouse, once it has been established.

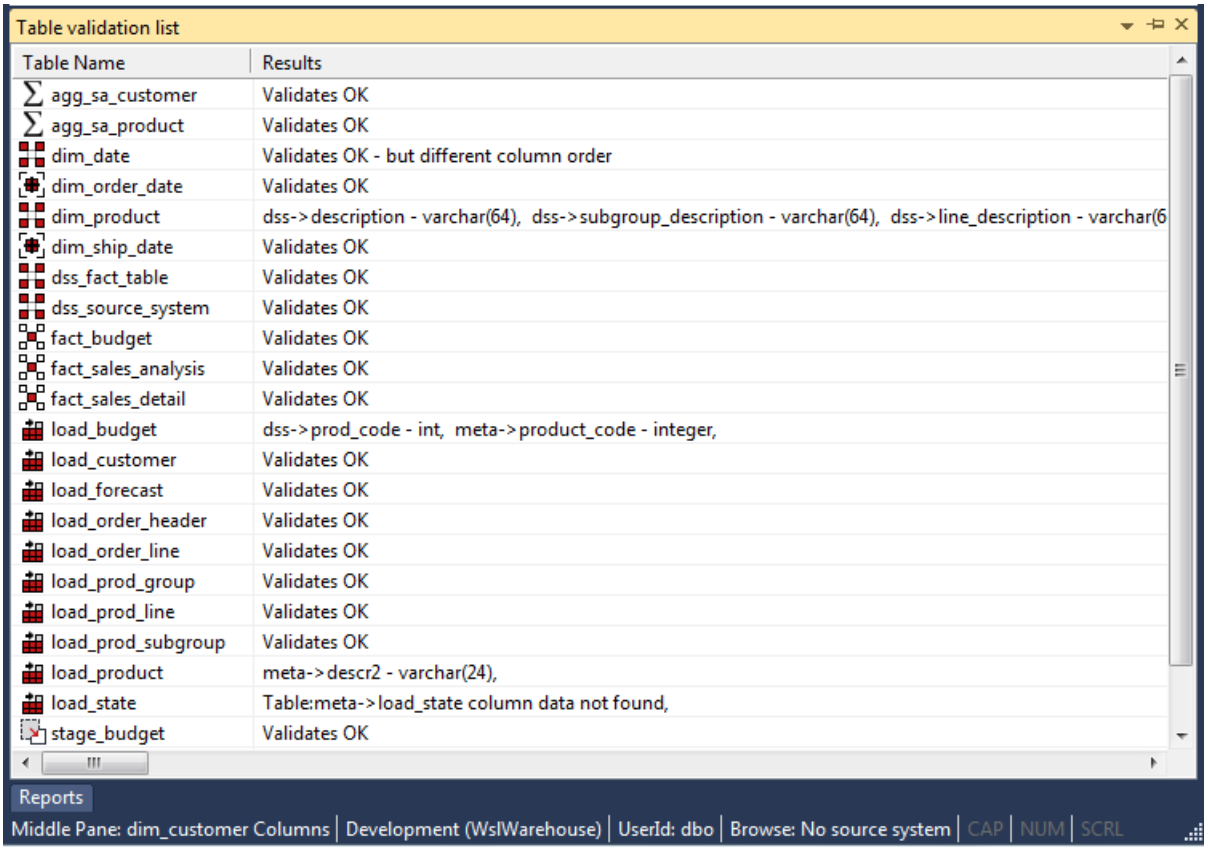
New source columns or changes to the source systems from which the data warehouse is built will require modifications to both the metadata and the data warehouse tables and procedures.

Validating Tables

The metadata as stored and maintained by WhereScape RED does not necessarily reflect the actual tables and procedures in use in the data warehouse. For example, if a new column is added to the metadata for a table, then that change is not automatically made in the actual physical table residing in the data warehouse. Likewise, if a column is deleted from the metadata, then that column may still exist in the physical database table.

This situation may be particularly apparent after an application patch or upgrade. The menu option **Validate > Validate Table Create Status**, and the right-click menu options in either the left or middle panes all provide a means of comparing the metadata to the physical tables in the database. A table, range of tables or all tables can be chosen. Each chosen table is a table in the metadata, and it is compared against the physical database table, if it exists.

The following example is the output from a validation.



In this example, we see five different scenarios.

1. The metadata for table **agg_sa_customer** matches the physical table in the database.
2. The table **dim_date** has the same columns in both the metadata and the physical table, but the column order is different. This is probably not an issue for most tables but may be a problem for some type of load tables, where the column order is important. This could be the result of a previous altering of the table. The table must be re-created if the order is important.
3. The physical database table **dim_product** have additional columns not found in the metadata. The columns are 'subgroup_description' and 'line_description'. The table can be altered if desired. See the next section on Altering Tables.

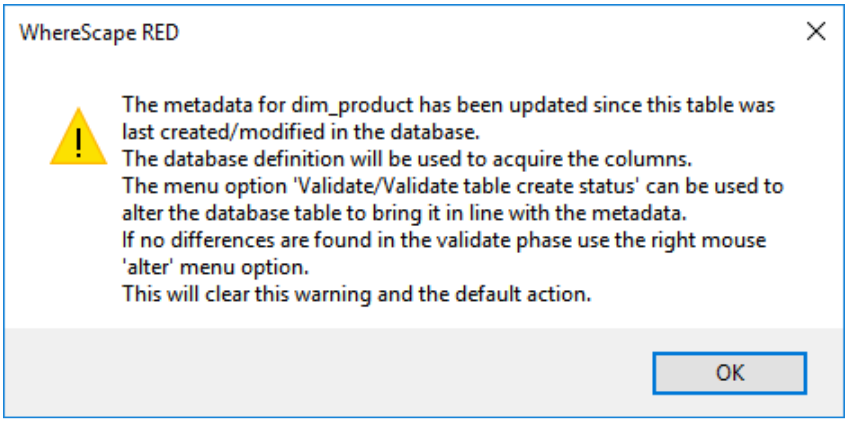
© Copyright, WhereScape Software Ltd All rights reserved.

4. The metadata for the table **load_product** does not match the physical table. The metadata has an additional column called 'state'. This column was not found in the physical table. The table can be altered if desired. See **Altering Tables** for details.
5. The table **load_state** is defined in the metadata but has not been physically created in the database. The table can be created in the normal manner.

Using outdated metadata in drag and drop

When dragging from a data warehouse table to create another data warehouse table (e.g. Load table to create Dimension table) a check is made to ensure that the metadata matches the database table.

If the two are found to be out of sync, the following message appears:

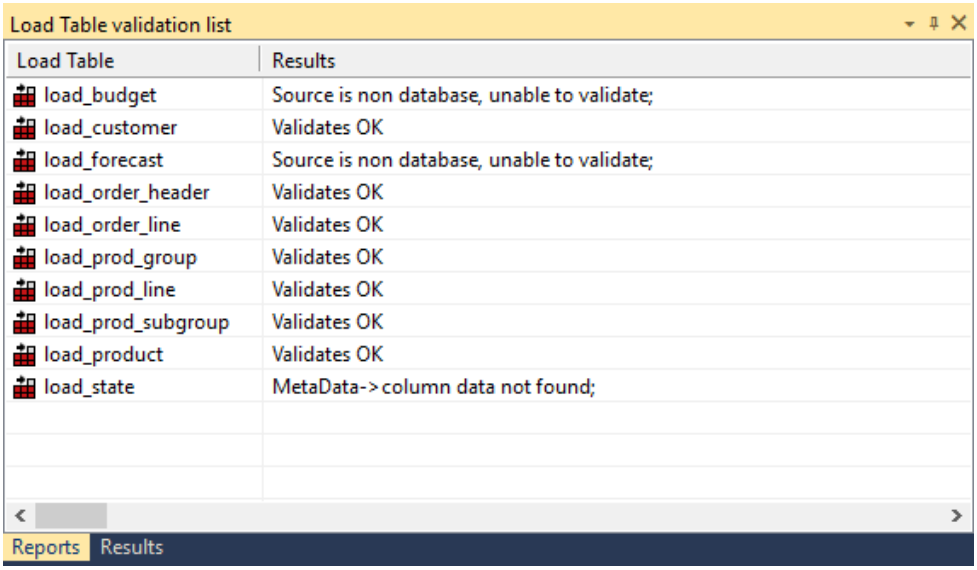


If a subsequent validate of the table in question shows that it validates, this message means that the dates are somehow out of sync. This can occur for example, after an import where the metadata has been replaced, but the underlying table still matches the metadata. Another common occurrence is where a new column is added and then deleted. To prevent the message from reoccurring in such a situation, proceed as follows.

Use the right-click menu and select **Alter Table** when positioned on the table name in the validate results screen (event though the table validates OK). The metadata update time is set back to that of the last database table create.

Validating Source (Load) Tables

Changes to the source systems from which the data warehouse is built can be detected to a limited degree. The menu option **Validate > Validate Load Table Status** enables a comparison between load tables and the source tables from which they were built. This comparison is not available for flat file or script-based loads. A Load table or group of load tables are selected, and the results are displayed in the middle pane. An example screen from a Load table validate is as follows:



The tables **load_budget** and **load_forecast** are Windows file loads and as such, cannot be validated.

© Copyright, WhereScape Software Ltd All rights reserved.

If a table shows additional columns in the source table; such a scenario will not cause problems for the continued operation of the data warehouse. It simply means that more columns are present in the source table than have been loaded into the data warehouse. This may have been the result of an initial partial selection or as a result of new columns. Further investigation of the source table would be required to ascertain if there was new information available.

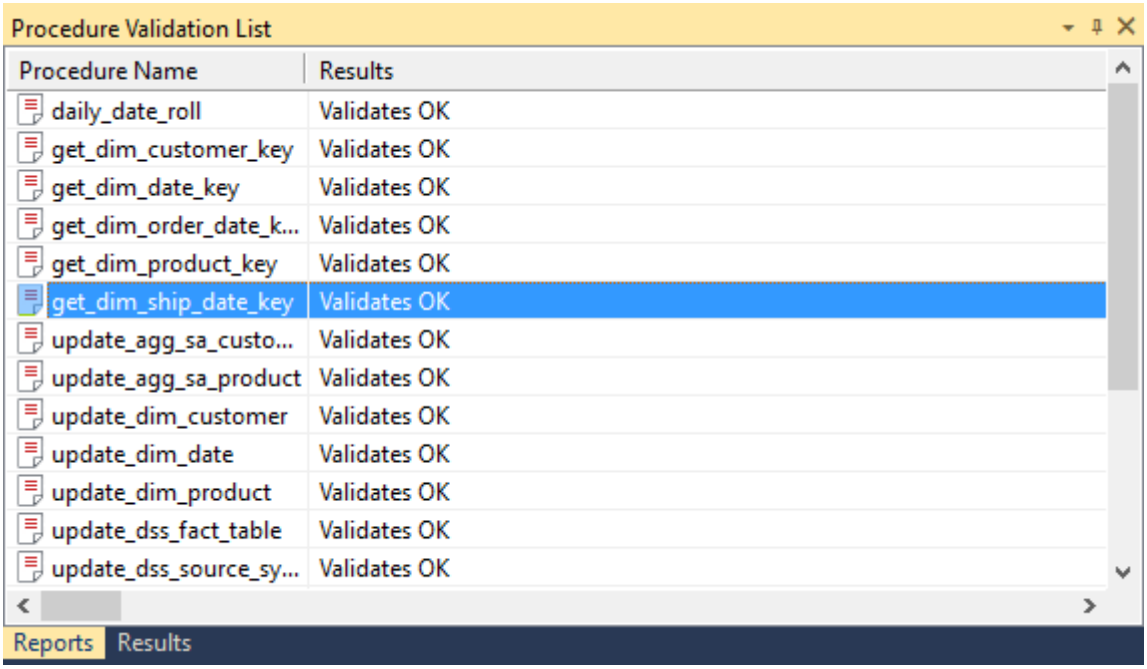
The table **load_state** reflects a problem for the continued operation of the data warehouse. The source table does not have a column that was previously identified as having come from that table. This will probably cause the load of that table to fail. This scenario would also require an investigation into the source table. The resolution may be to delete the column. The potential impact on later tables (Dimension, Stage and Fact) and procedures in the data warehouse can be ascertained by using the right-click menu of a Load table.

Validating Procedures

The menu option **Validate > Validate Procedure Status** compares procedures as stored in the metadata with those compiled and running in the data warehouse. This option provides a listing in the middle pane of each selected procedure and its status. The status will be **Validates OK**, **Not compiled**, or **Compare failed**.

Where a procedure is marked as **Not compiled**, this means that the procedure exists in the metadata but has not been compiled into the database.

Where a procedure fails to compare, the Procedure Editor must be used to find the actual differences. Start the editor by double-clicking the procedure object from the Objects list in the left pane. Use the **Tools > Compare to Compiled Source** menu option, to display the differences between the procedure in the metadata and that compiled in the database.



© Copyright, WhereScape Software Ltd All rights reserved.

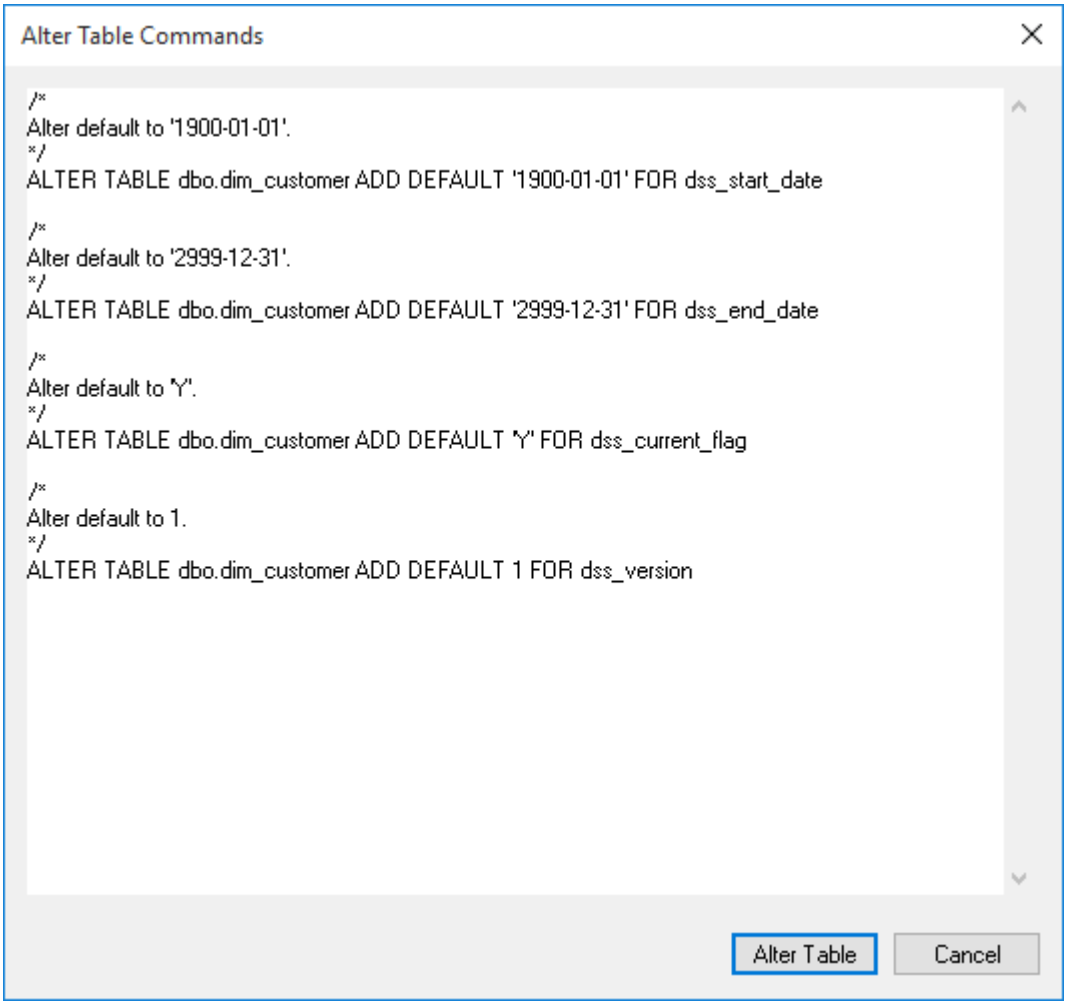
Altering Tables

The previous section covered the process of validating a table as defined in the metadata with the physical table as defined in the database. If a table is found to be different it is possible to alter the table.

Note

Caution should be taken when altering large data warehouse tables. A number of factors such as the time required to perform the alter, access to the table and the optimum storage of the table come into play.

To alter a table, first validate the table through the **Validate > Validate Table Create Status** menu option, or use the right-click menu option from the object name. Then in the middle pane (the validation listing) right-click the table that has not validated and select **Alter table**. A screen similar to the one below appears advising of the planned changes.



In this example, the **dim_product** table is to be altered. Comments at the top of the screen, show the reason(s) for the alteration and the actual alter command(s) follow.

The alter table window is an edit window. The command to be executed can be changed or additional commands entered. The command may also be cut to be executed in some other environment or at some later stage.

Clicking the **Alter Table** button alters the table. In effect, it will execute any command in the window.

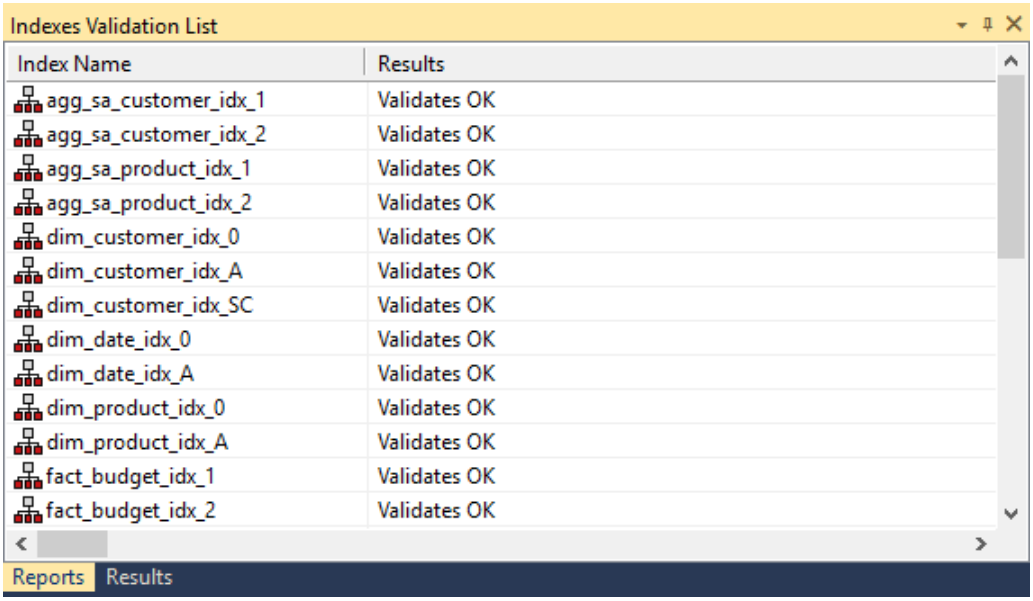
© Copyright, WhereScape Software Ltd All rights reserved.

Validating Indexes

The menu option **Validate/Validate Index Status** checks the metadata definition of an index against that in use in the database. It checks to ensure that the index exists and that the index type and columns are the same.

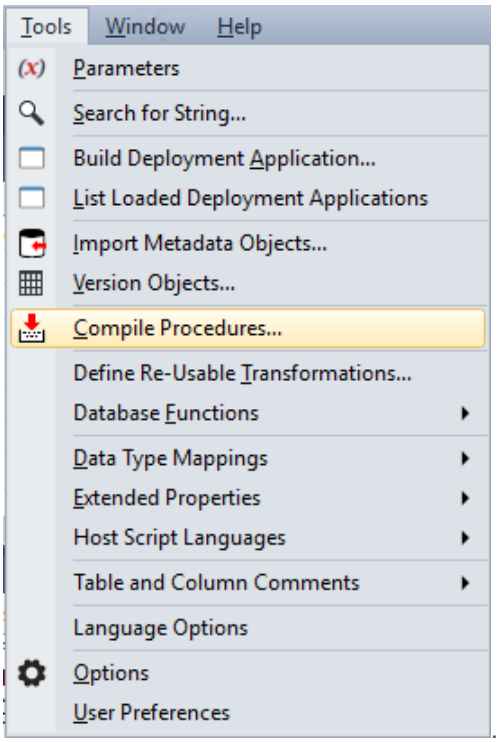
The results are listed in the middle pane with the status of each selected index shown.

If an index differs, then the only option is to drop and create the index either via the scheduler or through the right-click menu options for the index.



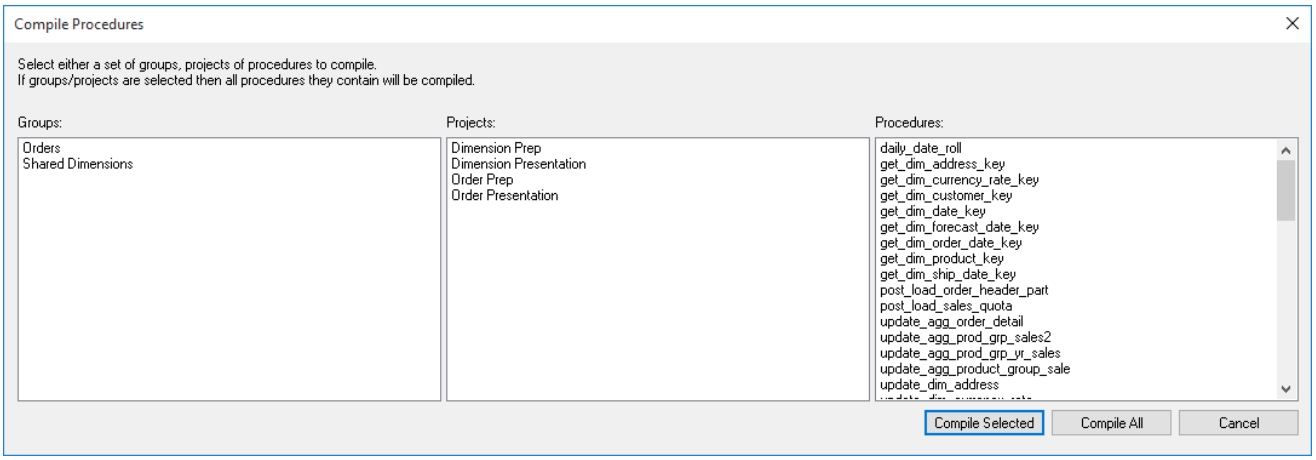
Recompiling Procedures

Procedures can be invalidated as a result of changes to underlying tables, or child procedures. A procedure can be recompiled through the procedure editor or via the menu option **Tools > Compile Procedures**:



© Copyright, WhereScape Software Ltd All rights reserved.

The **Compile Procedures** window is displayed :



All procedures in the metadata can be compiled, or a selected group of procedures may be compiled. Selection is done via standard Windows multi selection, using the shift and CTRL keys. By selecting a project or group it is possible to compile all procedures associated with the project or group.

As each procedure is compiled, it is displayed in the middle pane of the Builder window. If a procedure fails to compile, a failure message is displayed along side the procedure name. Procedures that fail to compile will need to be investigated through the procedure editor, as no specific error information is provided by this bulk compiler.

Re-validate Procedures (Oracle only)

The menu option **Tools > Re-compile Invalid Procedures** will attempt to validate the procedure by issuing an 'alter procedure xxx compile' command. This command will be executed for any procedure or function that is in an invalid state in the schema. Any procedures or functions that are still invalid after the commands have been completed are shown in the results pane. WhereScape RED does not drop procedures from the database, even when their metadata is deleted. Therefore, there may be old procedures present that are in an invalid state. These procedures may be redundant.

© Copyright, WhereScape Software Ltd All rights reserved.

Callable Routines

Introduction to Callable Routines

Callable Routines API

WhereScape RED callable routines provide an Application Program Interface (API) to the WhereScape RED metadata using the following SQL-invoked routines:

Routine Name	Description
Ws_Api_Glossary	Adds an entry to the documentation glossary.
Ws_Connect_Replace	Replaces the contents of a connection with details from another connection.
Ws_Job_Abort	Aborts a job if it is in a running state.
Ws_Job_Clear_Archive	Purges archived job logs that are older than the specified age in days.
Ws_Job_Clear_Logs	Archives job logs when the maximum number of current logs to retain is exceeded.
Ws_Job_Clear_Logs_By_Date	Archives job logs that are older than the specified age in days.
Ws_Job_Create	Creates a job based on an existing job and optionally starts it immediately.
Ws_Job_CreateWait	Creates a job based on an existing job and schedules it to start later.
Ws_Job_Dependency	Adds or removes a child-to-parent dependency between two jobs to control the child job.
Ws_Job_Release	Starts a job if it is a holding or waiting state.
Ws_Job_Restart	Starts a job if it is in a failed state.
Ws_Job_Schedule	Schedules a job if it is in a holding or waiting state.
Ws_Job_Status	Returns the current status of a job.
Ws_Load_Change	Changes the Connection or Schema of a load table.
Ws_Maintain_Indexes	Drops and/or builds database indexes that are defines in the WhereScape RED metadata.
Ws_Version_Clear	Purges metadata versions for all objects that do not meet the specified retention criteria.
WsParameterRead	Returns the value and comment (for most RDBMS) of a WhereScape RED metadata Parameter.
WsParameterReadF	Returns the value of a WhereScape RED metadata Parameter. **[SQL Server only]**
WsParameterReadG	Returns the value of a "global" WhereScape RED metadata Parameter that relates to a load table.
WsParameterWrite	Updates the value and comment of a WhereScape RED metadata Parameter or creates it.
WsWrkAudit	Records a message in the Audit Log.
WsWrkAuditBulk	Records multiple messages in the Audit Log.
WsWrkError	Records a message in the Error/Detail Log.
WsWrkErrorBulk	Records multiple messages in the Error/Detail Log.
WsWrkTask	Updates row counts for a task in the Task Log.

© Copyright, WhereScape Software Ltd All rights reserved.

Callable Routines per RDBMS

Each WhereScape RED Callable Routine exists in the database as either a Stored Procedure or a User-defined Function that can be invoked from SQL. A Callable Procedure is invoked by a SQL call/execute statement and a Callable Function is invoked in a SQL SELECT statement or a value expression. The Callable Routines are typically implemented as **procedures in most database systems** due to RDBMS limitations of user-defined functions, when the WhereScape RED API was originally developed.

Routine Name	SQL Server	Teradata	Oracle	DB2
Ws_Api_Glossary	Procedure	Procedure	Function	Procedure
Ws_Connect_Replace	Procedure	Procedure	Function	Procedure
Ws_Job_Abort	Procedure	Procedure	Function	Procedure
Ws_Job_Clear_Archive	Procedure	Procedure	Function	Procedure
Ws_Job_Clear_Logs	Procedure	Procedure	Function	Procedure
Ws_Job_Clear_Logs_By_Date	Procedure	Procedure	Function	Procedure
Ws_Job_Create	Procedure	Procedure	Function	Procedure
Ws_Job_CreateWait	Procedure	Procedure	Function	Procedure
Ws_Job_Dependency	Procedure	Procedure	Function	Procedure
Ws_Job_Release	Procedure	Procedure	Function	Procedure
Ws_Job_Restart	Procedure	Procedure	Function	Procedure
Ws_Job_Schedule	Procedure	Procedure	Function	Procedure
Ws_Job_Status	Procedure	Procedure	Function	Procedure
Ws_Load_Change	Procedure	Procedure	Function	Procedure
Ws_Maintain_Indexes	Procedure	Procedure	Function	Procedure
Ws_Version_Clear	Procedure	Procedure	Function	Procedure
WsParameterRead	Procedure	Procedure	Function	Procedure
WsParameterReadF	Procedure	Procedure	Function	Procedure
WsParameterReadG	Procedure	Procedure	Function	Procedure
WsParameterWrite	Procedure	Procedure	Function	Procedure
WsWrkAudit	Procedure	Procedure	Function	Procedure
WsWrkAuditBulk	Procedure	Procedure	Function	Procedure
WsWrkError	Procedure	Procedure	Function	Procedure
WsWrkErrorBulk	Procedure	Procedure	Function	Procedure
WsWrkTask	Procedure	Procedure	Function	Procedure

Callable Routines Names Qualifier

The WhereScape RED Callable Routines can be invoked using the unqualified routine name for SQL Server and Oracle. However, for Teradata and DB2 it is necessary to qualify the routine name with the owner/schema of the WhereScape RED metadata repository. All RED-generated procedures in a Teradata or DB2 repository that invoke a WhereScape RED Callable Routine do so by qualifying the routine name with **[METABASE]** e.g. **[METABASE]**.routine_name. When RED creates/compiles a procedure in a Teradata or DB2 database it automatically replaces the **[METABASE]** "token" with the repository owner/schema.

When you edit a RED-generated procedure or create your own custom procedure in Teradata or DB2, you can qualify each callable routine name by either hard-coding the owner/schema or by using the **[METABASE]** "token" that RED will replace when the procedure is created/compiled in the database. The Teradata and DB2 examples in this chapter use the **[METABASE]** "token" but if you invoke a WhereScape RED Callable Routine interactively or from outside a RED-compiled procedure/function, then the actual owner/schema must be specified because RED won't have the chance to replace the "token".

© Copyright, WhereScape Software Ltd All rights reserved.

Callable Routines Common Input

The following input parameters are common to most of the WhereScape RED Callable Routines, which are primarily used for integration with the WhereScape RED Scheduler.

Input	Parameter Name	Description
Job Instance Identifier	p_sequence	Unique identifier of the **running job** (i.e. the running instance of a held or scheduled job) that executed the routine. • When invoked from a WhereScape RED Scheduler the routine will be passed the parameter argument. • When invoked manually or externally to the WhereScape RED Scheduler, then any integer value can be used.
Job Name	p_job_name	Name of the **running job** that executed the routine. • When invoked from a WhereScape RED Scheduler the routine will be passed the parameter argument. • When invoked manually or externally to the WhereScape RED Scheduler, then any name can be used.
Task Name	p_task_name	Name of the **running task** (of a running job) that executed the routine. • When invoked from a WhereScape RED Scheduler the routine will be passed the parameter argument. • When invoked manually or externally to the WhereScape RED Scheduler, then any name can be used.
Job Identifier	p_job_id	Unique identifier of the **held or scheduled job** that the running job is a specific instance of. • When invoked from a WhereScape RED Scheduler the routine will be passed the parameter argument. • When invoked manually or externally to the WhereScape RED Scheduler, it is recommended to use 0 (zero).
Task Identifier	p_task_id	Unique identifier of the **running task** (of a running job) that executed the routine. • When invoked from a WhereScape RED Scheduler the routine will be passed the parameter argument. • When invoked manually or externally to the WhereScape RED Scheduler, it is recommended to use 0 (zero).

Note

Typically, the **parameter names** of the WhereScape RED Callable Routines use a p_ prefix as indicated but in some routines a v_ prefix is used instead. In addition, for SQL Server; all parameter names are also prefixed by an @ sign. The RDBMS-specific parameter names are included in the subsequent details of each Callable Routine.

Callable Routines Invocation

This chapter includes examples of each WhereScape RED Callable Routine to illustrate how to invoke the routine for each RDBMS along with the necessary variable declarations. Typically, the routines are executed from a Stored Procedure so the routine-specific examples illustrate that scenario and use parameters/arguments that are passed by position (as in a RED-generated procedure). However, it can be useful to execute a Callable Routine using other methods such as the following.

© Copyright, WhereScape Software Ltd All rights reserved.

Alternative Invocation Methods

Invocation using Named Parameters

SQL Server, Oracle, and DB2 support the passing of parameter arguments by **name** as an alternative to passing them by position. Typically, for most RDBMS when one parameter argument is supplied by name then all subsequent parameter arguments must be supplied in the same manner. The order of the parameter names doesn't matter but for clarity the following alternative examples pass the parameter names in the same order that they are declared in the routine. When parameter arguments are not passed by name the values must be supplied in the identical order (left to right) as the parameters are declared in the routine.

SQL Server example of invoking the Ws_Connect_Replace PROCEDURE using named parameters:

```
DECLARE @v_return_code varchar(1)
DECLARE @v_return_msg varchar(256)
DECLARE @v_result_num integer
EXEC Ws_Connect_Replace
@p_sequence = 0
, @p_job_name = 'Test Job Name'
, @p_task_name = 'Test Task Name'
, @p_job_id  = 0
, @p_task_id = 0
, @p_action = 'REPLACE'
, @p_source = 'Connection1'
, @p_target = 'Connection2'
, @p_return_code = @v_return_code OUTPUT
, @p_return_msg = @v_return_msg  OUTPUT
, @p_result = @v_result_num  OUTPUT
```

Oracle example of invoking the Ws_Connect_Replace FUNCTION using named parameters via an anonymous PL/SQL block:

```
DECLARE
v_return_code varchar2(1);
v_return_msg varchar2(256);
v_result_num integer;
BEGIN
v_result_num := Ws_Connect_Replace
( p_sequence    => 0
, p_job_name    => 'Test Job Name'
, p_task_name   => 'Test Task Name'
, p_job_id      => 0
, p_task_id     => 0
, p_action      => 'REPLACE'
, p_source      => 'Connection1'
, p_target      => 'Connection2'
, p_return_code => v_return_code
, p_return_msg  => v_return_msg
);
END;
/
```

© Copyright, WhereScape Software Ltd All rights reserved.

DB2 example of invoking the Ws_Connect_Replace PROCEDURE using named parameters via another procedure:

```
-- The owner/schema is DSSDEMO.
DROP PROCEDURE DSSDEMO.invoke_DB2_Ws_Connect_Replace;
CREATE PROCEDURE DSSDEMO.invoke_DB2_Ws_Connect_Replace
LANGUAGE SQL
BEGIN
DECLARE v_return_code           varchar(1);
DECLARE v_return_msg            varchar(256);
DECLARE v_result_num            integer;
CALL DSSDEMO.Ws_Connect_Replace
( p_sequence    => 0
, p_job_name    => 'Test Job Name'
, p_task_name   => 'Test Task Name'
, p_job_id      => 0
, p_task_id     => 0
, p_action      => 'REPLACE'
, p_source      => 'Connection1'
, p_target      => 'Connection2'
, p_return_code => v_return_code
, p_return_msg  => v_return_msg
, p_result      => v_result_num
);
END
/
CALL DSSDEMO.invoke_DB2_Ws_Connect_Replace();
```

Invocation via ODBC

The following examples illustrate how to invoke a WhereScape RED Callable Routine via an ODBC connection (using a tool, such as WhereScape SQL Admin), which uses both input and output parameters. The examples work for a SQL Server RED repository but for another RDBMS the routine name may need to be qualified with the appropriate owner/schema.
Refer to the detailed description of the **Ws_Connect_Replace** routine for an explanation of the parameters/arguments.

The following invocations via ODBC are equivalent and the only difference is the formatting as the first command uses a single line while the second command is formatted across multiple lines:

-- OBDC Example 1 (single line).

```
{ CALL Ws_Connect_Replace( 0,'Test Job Name', 'Test Task Name', 0, 0, 'REPLACE',
'Connection1', 'Connection2', ?, ?, ?) };
```

-- OBDC Example 2 (same command formatted using multiple lines).

```
{ CALL Ws_Connect_Replace
( 0
, 'Test Job Name'
, 'Test Task Name'
, 0
, 0
, 'REPLACE'
, 'Connection1'
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
, 'Connection2'
, ?
, ?
, ?
)
};
```

The result of the Ws_Connect_Replace invocation can be confirmed by checking the target connection. In addition, a log entry is created using the specified job and task names that can be viewed via the Logs—Recent Audit Trail Logs menu item of the WhereScape RED Scheduler.

Invocation via the Command-Line

Each WhereScape RED Callable Routine can also be invoked from the command-line using an RDBMS-specific tool such as:

RDBMS	SQL Command-Line-Interface (CLI) Tool
SQL Server	SQL Server sqlcmd Utility i.e. sqlcmd.exe.
Teradata	Basic Teradata Query (BTEQ) i.e. bteq.
Oracle	Oracle SQL*Plus i.e. sqlplus.

These tools can be used to invoke a WhereScape RED Callable Routine by connecting to the database, executing SQL statements, and disconnecting from the database. Refer to the RDBMS-specific documentation for details of how to connect and execute SQL via the relevant CLI tool as well as details of tool's return codes.

Typically, the CLI command will include options to specify the database, connection credentials, and the SQL statement to execute. Multiple SQL statements can typically be executed by terminating/delimiting each statement, using a semi-colon (;). In most cases, the SQL statement needs to be quoted (typically double-quoted) when specified on the command-line and if the SQL statements include embedded quotes (such as single-quotes around literals) then they may need to be "escaped" depending on the CLI tool and the platform. Most of the tools also allow the SQL commands to be read from a file instead of from standard input.

Ws_Api_Glossary

Synopsis

Adds an entry to the documentation glossary.

Description

Adds the specified entry to the documentation glossary, which is included in documentation that is subsequently generated by WhereScape RED.

Input

Input	Description
Object Name	Item that appears in the left column of the glossary, which normally represents the business name of a column in a dimension or fact table (although it can be used for other purposes).
Glossary Term	Item that appears under the analysis area heading of the glossary, which normally represents a dimension ot fact table name (although it can be used for other purposes).
Glossary Description	Description of the term being defined.
Action	Either **ADD** or **DELETE** the specified glossary entry.

Output

Output	Description
Result Text	A message to indicate whether or not the glossary term was successfully added/deleted.

© Copyright, WhereScape Software Ltd All rights reserved.

SQL Server

SQL Server Parameters: Ws_Api_Glossary
Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
@p_object_name	VARCHAR(64)	IN
@p_term	VARCHAR(256)	IN
@p_comments	VARCHAR(4000)	IN
@p_option	VARCHAR(12)	IN
@p_result	VARCHAR(256)	OUT

SQL Server Examples: Ws_Api_Glossary

```
DECLARE @v_result_txt varchar(256)
EXEC Ws_Api_Glossary
  'Data Warehouse'
, 'Overview'
, 'A repository of business information'
, 'ADD'
, @v_result_txt OUTPUT
```

Teradata

Teradata Parameters: Ws_Api_Glossary
Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_object_name	VARCHAR(64)	IN
p_term	VARCHAR(256)	IN
p_comments	VARCHAR(4000)	IN
p_option	VARCHAR(64)	IN
p_result	VARCHAR(256)	OUT

Teradata Examples: Ws_Api_Glossary

```
DECLARE v_result_txt varchar(256);
CALL [METABASE].Ws_Api_Glossary
( 'Data Warehouse'
, 'Overview'
, 'A repository of business information'
, 'ADD'
, v_result_txt
);
```

Oracle

Oracle Parameters: Ws_Api_Glossary
Callable Routine Type: FUNCTION.

Parameter Name	Datatype	Mode
v_object_name	VARCHAR2	IN

© Copyright, WhereScape Software Ltd All rights reserved.

Parameter Name	Datatype	Mode
v_term	VARCHAR2	IN
v_comments	VARCHAR2	IN
v_option	VARCHAR2	IN
FUNCTION Return Value	VARCHAR2	OUT-Function

Oracle Examples: Ws_Api_Glossary

```
v_result_txt  varchar2(256);
v_result_txt := Ws_Api_Glossary
( 'Data Warehouse'
, 'Overview'
, 'A repository of business information'
, 'ADD'
);
```

DB2

DB2 Parameters: Ws_Api_Glossary

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_object_name	VARCHAR(64)	IN
p_term	VARCHAR(256)	IN
p_comments	VARCHAR(4000)	IN
p_option	VARCHAR(64)	IN
p_result	VARCHAR(256)	OUT

DB2 Examples: Ws_Api_Glossary

```
DECLARE v_result_txt varchar(256);
CALL [METABASE].Ws_Api_Glossary
( 'Data Warehouse'
, 'Overview'
, 'A repository of business information'
, 'ADD'
, v_result_txt
);
```

Ws_Connect_Replace

Synopsis

Replaces the contents of a connection with the details from another connection.

Description

Copies the details of the specified Source connection to the specified Target connection.

Input

Input	Description
Common Input	Includes all 5 inputs of the **Callable Routines Common Input**.

© Copyright, WhereScape Software Ltd All rights reserved.

Input	Description
Action	**REPLACE** the details of the specified Target connection.
Source Connection	The name of the Source connection whose details will be copied.
Target Connection	The name of the Target connection whose details will be changed.

Output

Output	Description
Return Code	Output Return Code: • S Success. • E Error. • F Fatal/Unexpected Error.
Return Message	Output message indicating the action applied or the reason for no action.
Result Number	Output Result Number: • 1 Success. • -2 Error. • -3 Fatal/Unexpected Error.

SQL Server

SQL Server Parameters: Ws_Connect_Replace

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
@p_sequence	INTEGER	IN
@p_job_name	VARCHAR(64)	IN
@p_task_name	VARCHAR(64)	IN
@p_job_id	INTEGER	IN
@p_task_id	INTEGER	IN
@p_action	VARCHAR(64)	IN
@p_source	VARCHAR(64)	IN
@p_target	VARCHAR(64)	IN
@p_return_code	VARCHAR(1)	OUT
@p_return_msg	VARCHAR(256)	OUT
@p_result	INTEGER	OUT

SQL Server Examples: Ws_Connect_Replace

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE @p_sequence integer
DECLARE @p_job_name varchar(256)
DECLARE @p_task_name varchar(256)
DECLARE @p_job_id integer
DECLARE @p_task_id integer
DECLARE @p_return_msg varchar(256)
DECLARE @p_status  integer
DECLARE @v_result_num integer
DECLARE @v_return_code varchar(1)
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
DECLARE @v_return_msg varchar(256)
EXEC Ws_Connect_Replace
@p_sequence, @p_job_name, @p_task_name, @p_job_id, @p_task_id
, 'REPLACE', 'Connection1', 'Connection2'
, @v_return_code OUTPUT
, @v_return_msg  OUTPUT
, @v_result_num  OUTPUT
```

Teradata

Teradata Parameters: Ws_Connect_Replace

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_action	VARCHAR(64)	IN
p_source	VARCHAR(64)	IN
p_target	VARCHAR(64)	IN
p_return_code	VARCHAR(1)	OUT
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT

Teradata Examples: Ws_Connect_Replace

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_sequence integer;
DECLARE p_job_name varchar(256);
DECLARE p_task_name varchar(256);
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE p_return_msg varchar(256);
DECLARE p_status integer;
DECLARE v_result_num integer;
DECLARE v_return_code varchar(1);
DECLARE v_return_msg varchar(256);
CALL [METABASE].Ws_Connect_Replace
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, 'REPLACE', 'Connection1', 'Connection2'
, v_return_code
, v_return_msg
, v_result_num
);
```

© Copyright, WhereScape Software Ltd All rights reserved.

Oracle

Oracle Parameters: Ws_Connect_Replace

Callable Routine Type: FUNCTION.

Parameter Name	Datatype	Mode
p_sequence	NUMBER	IN
p_job_name	VARCHAR2	IN
p_task_name	VARCHAR2	IN
p_job_id	NUMBER	IN
p_task_id	NUMBER	IN
p_action	VARCHAR2	IN
p_source	VARCHAR2	IN
p_target	VARCHAR2	IN
p_return_code	VARCHAR2	OUT
p_return_msg	VARCHAR2	OUT
FUNCTION Return Value	NUMBER	OUT-Function

Oracle Examples: Ws_Connect_Replace

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
p_sequence number;
p_job_name varchar2(256);
p_task_name varchar2(256);
p_job_id number;
p_task_id number;
p_return_msg varchar2(256);
p_status number;
v_result_num number;
v_return_code varchar2(1);
v_return_msg varchar2(256);
v_result_num := Ws_Connect_Replace
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, 'REPLACE', 'Connection1', 'Connection2'
, v_return_code
, v_return_msg
);
```

DB2

DB2 Parameters: Ws_Connect_Replace

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN

© Copyright, WhereScape Software Ltd All rights reserved.

Parameter Name	Datatype	Mode
p_action	VARCHAR(64)	IN
p_source	VARCHAR(64)	IN
p_target	VARCHAR(64)	IN
p_return_code	VARCHAR(1)	OUT
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT

DB2 Examples: Ws_Connect_Replace

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_sequence integer;
DECLARE p_job_name varchar(256);
DECLARE p_task_name varchar(256);
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE p_return_msg varchar(256);
DECLARE p_status integer;
DECLARE v_result_num integer;
DECLARE v_return_code varchar(1);
DECLARE v_return_msg varchar(256);
CALL [METABASE].Ws_Connect_Replace
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, 'REPLACE', 'Connection1', 'Connection2'
, v_return_code
, v_return_msg
, v_result_num
);
```

Ws_Job_Abort

Synopsis

Aborts a job if it is in a running state.

Description

Aborts the specified job if it is in a running state, which changes it to a failed state, fails all running tasks, and holds all waiting tasks.

Input

Input	Description
Abort Job Name	The name of the job to be aborted. The specified name must exactly match the job name as displayed by the WhereScape RED Scheduler. The specified job must be in a running or failed state in order to be aborted.
Job Instance Identifier	Unique identifier of the running job (that may be in a failed state).
Abort Job Message Text	Custom message text to be recorded in the WhereScape RED Audit Log for the aborted job and the WhereScape RED Task Log for each aborted task.

© Copyright, WhereScape Software Ltd All rights reserved.

Output

Output	Description
Result Number[ORACLE only]	Output Result Number [ORACLE only]: • 1 Success. • -2 Error. e.g. Due to invalid job name or job not running. • -3 Fatal/Unexpected Error.

SQL Server

SQL Server Parameters: Ws_Job_Abort

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
@p_job_name	VARCHAR(64)	IN
@p_job_sequence	INTEGER	IN
@p_job_msg	VARCHAR(256)	IN

SQL Server Examples: Ws_Job_Abort

```
EXEC Ws_Job_Abort
  'Daily Run'
, 1234
, 'Job aborted via manual execution of Ws_Job_Abort.'
```

Teradata

Teradata Parameters: Ws_Job_Abort

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_job_name	VARCHAR(64)	IN
p_job_sequence	INTEGER	IN
p_job_msg	VARCHAR(256)	IN

Teradata Examples: Ws_Job_Abort

```
CALL [METABASE].Ws_Job_Abort
( 'Daily Run'
, 1234
, 'Job aborted via manual execution of Ws_Job_Abort.'
);
```

Oracle

Oracle Parameters: Ws_Job_Abort

Callable Routine Type: FUNCTION.

Parameter Name	Datatype	Mode
v_job_name	VARCHAR2	IN
v_job_sequence	NUMBER	IN
v_job_msg	VARCHAR2	IN

© Copyright, WhereScape Software Ltd All rights reserved.

Parameter Name	Datatype	Mode
FUNCTION Return Value	NUMBER	OUT-Function

Oracle Examples: Ws_Job_Abort

```
v_result_num                    number;
v_result_num := Ws_Job_Abort
( 'Daily Run'
, 1234
, 'Job aborted via manual execution of Ws_Job_Abort.'
);
```

DB2

DB2 Parameters: Ws_Job_Abort

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_job_name	VARCHAR(64)	IN
p_job_sequence	INTEGER	IN
p_job_msg	VARCHAR(256)	IN

DB2 Examples: Ws_Job_Abort

```
CALL [METABASE].Ws_Job_Abort
( 'Daily Run'
, 1234
, 'Job aborted via manual execution of Ws_Job_Abort.'
);
```

Ws_Job_Clear_Archive

Synopsis

Purges archived job logs that are older than the specified age in days.

Description

Deletes job-related logs that were previously archived (into the WX_WRK_AUDIT_ARCHIVE and WX_WRK_ERROR_ARCHIVE tables via a RED Scheduler and/or RED callable routines, such as Ws_Job_Clear_Logs and Ws_Job_Clear_Logs_By_Date) depending on their age in days.

When the maximum age of the archived logs to retain is exceeded all the older logs are deleted. For example, if 90 days are retained then all the archived logs that are older than 90 days are deleted. If a maximum age of 0 days is specified then all the archived logs are deleted. Alternatively, the **TRUNCATE** option can be used to remove all the archived logs, which overrides all other criteria.

Input

Input	Description
Common Input	Includes all 5 inputs of the **Callable Routines Common Input**.
Maximum Days to Retain	The maximum age (in days) of the archived logs to retain. If 90 days are retained then all the archived logs that are older than 90 days are **purged/deleted**. If 0 days are retained then all the archived logs are **purged/deleted**.
Job Name to Purge	The name of the job whose **archived** logs are to be **purged**. Wild cards are supported. Specifying % will match ALL jobs.

© Copyright, WhereScape Software Ltd All rights reserved.

Input	Description
Options	The **TRUNCATE** option can be used to remove ALL the archived logs, which overrides all other criteria i.e. irrespective of the days to retain or job name.

Output

Output	Description
Return Code	Output Return Code: • S Success. • E Error. • F Fatal/Unexpected Error.
Return Message	Output message indicating the action applied or the reason for no action.
Result Number	Output Result Number: • 1 Success. • -2 Error. e.g. Due to invalid job name or job not running. • -3 Fatal/Unexpected Error.

SQL Server

SQL Server Parameters: Ws_Job_Clear_Archive

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
@p_sequence	INTEGER	IN
@p_job_name	VARCHAR(64)	IN
@p_task_name	VARCHAR(64)	IN
@p_job_id	INTEGER	IN
@p_task_id	INTEGER	IN
@p_day_count	INTEGER	IN
@p_job	VARCHAR(64)	IN
@p_options	VARCHAR(256)	IN
@p_return_code	VARCHAR(1)	OUT
@p_return_msg	VARCHAR(256)	OUT
@p_result	INTEGER	OUT

SQL Server Examples: Ws_Job_Clear_Archive

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE @p_sequence integer
DECLARE @p_job_name varchar(256)
DECLARE @p_task_name varchar(256)
DECLARE @p_job_id integer
DECLARE @p_task_id  integer
DECLARE @p_return_msg varchar(256)
DECLARE @p_status integer
DECLARE @v_result_num integer
DECLARE @v_return_code varchar(1)
DECLARE @v_return_msg varchar(256)
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
EXEC Ws_Job_Clear_Archive
@p_sequence, @p_job_name, @p_task_name, @p_job_id, @p_task_id
, 90, 'Daily Run', ''
, @v_return_code OUTPUT
, @v_return_msg  OUTPUT
, @v_result_num  OUTPUT
```

Teradata

Teradata Parameters: Ws_Job_Clear_Archive
Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_day_count	INTEGER	IN
p_job	VARCHAR(64)	IN
p_options	VARCHAR(256)	IN
p_return_code	VARCHAR(1)	OUT
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT

Teradata Examples: Ws_Job_Clear_Archive

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_sequence integer;
DECLARE p_job_name varchar(256);
DECLARE p_task_name varchar(256);
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE p_return_msg varchar(256);
DECLARE p_status integer;
DECLARE v_result_num integer;
DECLARE v_return_code varchar(1);
DECLARE v_return_msg varchar(256);
CALL [METABASE].Ws_Job_Clear_Archive
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, 90, 'Daily Run', ''
, v_return_code
, v_return_msg
, v_result_num
);
```

© Copyright, WhereScape Software Ltd All rights reserved.

Oracle

Oracle Parameters: Ws_Job_Clear_Archive

Callable Routine Type: FUNCTION.

Parameter Name	Datatype	Mode
p_sequence	NUMBER	IN
p_job_name	VARCHAR2	IN
p_task_name	VARCHAR2	IN
p_job_id	NUMBER	IN
p_task_id	NUMBER	IN
p_day_count	NUMBER	IN
p_job	VARCHAR2	IN
p_options	VARCHAR2	IN
p_return_code	VARCHAR2	OUT
p_return_msg	VARCHAR2	OUT
FUNCTION Return Value	NUMBER	OUT-Function

Oracle Examples: Ws_Job_Clear_Archive

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
p_sequence number;
p_job_name varchar2(256);
p_task_name varchar2(256);
p_job_id number;
p_task_id number;
p_return_msg varchar2(256);
p_status number;
v_result_num number;
v_return_code varchar2(1);
v_return_msg varchar2(256);
v_result_num := Ws_Job_Clear_Archive
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, 90, 'Daily Run', ''
, v_return_code
, v_return_msg
);
```

DB2

DB2 Parameters: Ws_Job_Clear_Archive

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN

© Copyright, WhereScape Software Ltd All rights reserved.

Parameter Name	Datatype	Mode
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_day_count	INTEGER	IN
p_job	VARCHAR(64)	IN
p_options	VARCHAR(256)	IN
p_return_code	VARCHAR(1)	OUT
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT

DB2 Examples: Ws_Job_Clear_Archive

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_sequence integer;
DECLARE p_job_name varchar(256);
DECLARE p_task_name varchar(256);
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE p_return_msg varchar(256);
DECLARE p_status integer;
DECLARE v_result_num integer;
DECLARE v_return_code varchar(1);
DECLARE v_return_msg varchar(256);
CALL [METABASE].Ws_Job_Clear_Archive
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, 90, 'Daily Run', ''
, v_return_code
, v_return_msg
, v_result_num
);
```

© Copyright, WhereScape Software Ltd All rights reserved.

Ws_Job_Clear_Logs

Synopsis

Archives job logs when the maximum number of current logs to retain is exceeded.

Description

Moves job-related logs from the current log tables (such as WS_WRK_AUDIT_LOG and WS_WRK_ERROR_LOG) to the corresponding archive log tables (such as WX_WRK_AUDIT_ARCHIVE and WX_WRK_ERROR_ARCHIVE) depending on the number of logs to retain.

When the maximum number of current logs to retain is exceeded the oldest logs are archived for the specified job(s) to reduce the number of current logs to the specified retention limit. For example, if 10 is specified then only the latest 10 logs are retained. If a retained count of 0 is specified then all the current logs are archived for the specified job(s).

Note

Equivalent functionality is available via a WhereScape RED Scheduler and the "Logs Retained" property of a job.

Input

Input	Description
Common Input	Includes all 5 inputs of the **Callable Routines Common Input**.
Job Name(s) to Archive	The name of the job(s) whose current logs are to be archived. Wild cards are supported. Specifying % will match ALL jobs.
Maximum Logs to Retain	The maximum number of logs to retain. When the maximum is exceeded, the oldest logs are **archived** to reduce the number of current logs to the specified retention limit.

Output

Output	Description
Return Code	Output Return Code: • S Success. • E Error. • F Fatal/Unexpected Error.
Return Message	Output message indicating the action applied or the reason for no action.
Result Number	Output Result Number: • 1 Success. • -2 Error. e.g. Due to invalid job name or job not running. • -3 Fatal/Unexpected Error.

SQL Server

SQL Server Parameters: Ws_Job_Clear_Logs

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
@p_sequence	INTEGER	IN
@p_job_name	VARCHAR(64)	IN
@p_task_name	VARCHAR(64)	IN
@p_job_id	INTEGER	IN
@p_task_id	INTEGER	IN
@p_job_to_clean	VARCHAR(64)	IN

© Copyright, WhereScape Software Ltd All rights reserved.

Parameter Name	Datatype	Mode
@p_keep_count	INTEGER	IN
@p_return_code	VARCHAR(1)	OUT
@p_return_msg	VARCHAR(256)	OUT
@p_result	INTEGER	OUT

SQL Server Examples: Ws_Job_Clear_Logs

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE @p_sequence integer
DECLARE @p_job_name varchar(256)
DECLARE @p_task_name varchar(256)
DECLARE @p_job_id integer
DECLARE @p_task_id integer
DECLARE @p_return_msg varchar(256)
DECLARE @p_status integer
DECLARE @v_result_num integer
DECLARE @v_return_code varchar(1)
DECLARE @v_return_msg varchar(256)
EXEC Ws_Job_Clear_Logs
 @p_sequence, @p_job_name, @p_task_name, @p_job_id, @p_task_id
, 'Daily Run', 10
, @v_return_code OUTPUT
, @v_return_msg OUTPUT
, @v_result_num OUTPUT
```

Teradata

Teradata Parameters: Ws_Job_Clear_Logs
Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_job_to_clean	VARCHAR(64)	IN
p_log_keep	INTEGER	IN
p_return_code	VARCHAR(1)	OUT
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT

Teradata Examples: Ws_Job_Clear_Logs

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_sequence integer;
DECLARE p_job_name varchar(256);
DECLARE p_task_name varchar(256);
DECLARE p_job_id integer;
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
DECLARE p_task_id integer;
DECLARE p_return_msg varchar(256);
DECLARE p_status integer;
DECLARE v_result_num integer;
DECLARE v_return_code varchar(1);
DECLARE v_return_msg varchar(256);
CALL [METABASE].Ws_Job_Clear_Logs
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, 'Daily Run', 10
, v_return_code
, v_return_msg
, v_result_num
);
```

Oracle

Oracle Parameters: Ws_Job_Clear_Logs
Callable Routine Type: FUNCTION.

Parameter Name	Datatype	Mode
p_sequence	NUMBER	IN
p_job_name	VARCHAR2	IN
p_task_name	VARCHAR2	IN
p_job_id	NUMBER	IN
p_task_id	NUMBER	IN
p_job_to_clean	VARCHAR2	IN
p_log_keep	NUMBER	IN
p_return_code	VARCHAR2	OUT
p_return_msg	VARCHAR2	OUT
FUNCTION Return Value	NUMBER	OUT-Function

Oracle Examples: Ws_Job_Clear_Logs

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
p_sequence number;
p_job_name varchar2(256);
p_task_name varchar2(256);
p_job_id number;
p_task_id number;
p_return_msg varchar2(256);
p_status number;
v_result_num number;
v_return_code varchar2(1);
v_return_msg varchar2(256);
v_result_num := Ws_Job_Clear_Logs
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, 'Daily Run', 10
, v_return_code
, v_return_msg
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
);
```

DB2

DB2 Parameters: Ws_Job_Clear_Logs

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_job_to_clean	VARCHAR(64)	IN
p_keep_count	INTEGER	IN
p_return_code	VARCHAR(1)	OUT
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT

DB2 Examples: Ws_Job_Clear_Logs

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_sequence integer;
DECLARE p_job_name varchar(256);
DECLARE p_task_name varchar(256);
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE p_return_msg varchar(256);
DECLARE p_status integer;
DECLARE v_result_num integer;
DECLARE v_return_code varchar(1);
DECLARE v_return_msg varchar(256);
CALL [METABASE].Ws_Job_Clear_Logs
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, 'Daily Run', 10
, v_return_code
, v_return_msg
, v_result_num
);
```

© Copyright, WhereScape Software Ltd All rights reserved.

Ws_Job_Clear_Logs_By_Date

Synopsis

Archives job logs that are older than the specified age in days.

Description

Moves job-related logs from the current log tables (such as WS_WRK_AUDIT_LOG and WS_WRK_ERROR_LOG) to the corresponding archive log tables (such as WX_WRK_AUDIT_ARCHIVE and WX_WRK_ERROR_ARCHIVE) depending on their age in days.

When the maximum age of the current logs to retain is exceeded all the older logs are archived for the specified job(s). For example, if 90 days are retained then all the current logs that are older than 90 days are archived.

Input

Input	Description
Common Input	Includes all 5 inputs of the **Callable Routines Common Input**.
Job Name(s) to Archive	The name of the job(s) whose current logs are to be **archived**. Wild cards are supported. Specifying % will match ALL jobs.
Maximum Days to Retain	The maximum age (in days) of the current logs to retain. If 90 days are retained, then all the current logs that are older than 90 days are archived.

Output

Output	Description
Return Code	Output Return Code: • S Success. • E Error. • F Fatal/Unexpected Error.
Return Message	Output message indicating the action applied or the reason for no action.
Result Number	Output Result Number: • 1 Success. • -2 Error. e.g. Due to invalid job name or job not running. • -3 Fatal/Unexpected Error.

SQL Server

SQL Server Parameters: Ws_Job_Clear_Logs_By_Date

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
@p_sequence	INTEGER	IN
@p_job_name	VARCHAR(64)	IN
@p_task_name	VARCHAR(64)	IN
@p_job_id	INTEGER	IN
@p_task_id	INTEGER	IN
@p_job_to_clean	VARCHAR(64)	IN
@p_day_count	INTEGER	IN
@p_return_code	VARCHAR(1)	OUT
@p_return_msg	VARCHAR(256)	OUT
@p_result	INTEGER	OUT

© Copyright, WhereScape Software Ltd All rights reserved.

SQL Server Examples: Ws_Job_Clear_Logs_By_Date

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE @p_sequence integer
DECLARE @p_job_name varchar(256)
DECLARE @p_task_name varchar(256)
DECLARE @p_job_id integer
DECLARE @p_task_id integer
DECLARE @p_return_msg varchar(256)
DECLARE @p_status integer
DECLARE @v_result_num integer
DECLARE @v_return_code varchar(1)
DECLARE @v_return_msg varchar(256)
EXEC Ws_Job_Clear_Logs_By_Date
@p_sequence, @p_job_name, @p_task_name, @p_job_id, @p_task_id
, 'Daily Run', 90
, @v_return_code OUTPUT
, @v_return_msg OUTPUT
, @v_result_num OUTPUT
```

Teradata

Teradata Parameters: Ws_Job_Clear_Logs_By_Date
Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_job_to_clean	VARCHAR(64)	IN
p_day_count	INTEGER	IN
p_return_code	VARCHAR(1)	OUT
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT

Teradata Examples: Ws_Job_Clear_Logs_By_Date

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_sequence integer;
DECLARE p_job_name varchar(256);
DECLARE p_task_name varchar(256);
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE p_return_msg varchar(256);
DECLARE p_status integer;
DECLARE v_result_num integer;
DECLARE v_return_code varchar(1);
DECLARE v_return_msg varchar(256);
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
CALL [METABASE].Ws_Job_Clear_Logs_By_Date
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, 'Daily Run', 90
, v_return_code
, v_return_msg
, v_result_num
);
```

Oracle

Oracle Parameters: Ws_Job_Clear_Logs_By_Date
Callable Routine Type: FUNCTION.

Parameter Name	Datatype	Mode
p_sequence	NUMBER	IN
p_job_name	VARCHAR2	IN
p_task_name	VARCHAR2	IN
p_job_id	NUMBER	IN
p_task_id	NUMBER	IN
p_job_to_clean	VARCHAR2	IN
p_day_count	NUMBER	IN
p_return_code	VARCHAR2	OUT
p_return_msg	VARCHAR2	OUT
p_result	NUMBER	OUT-Function

Oracle Examples: Ws_Job_Clear_Logs_By_Date

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
p_sequence number;
p_job_name varchar2(256);
p_task_name varchar2(256);
p_job_id number;
p_task_id number;
p_return_msg varchar2(256);
p_status number;
v_result_num number;
v_return_code varchar2(1);
v_return_msg varchar2(256);
v_result_num := Ws_Job_Clear_Logs_By_Date
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, 'Daily Run', 90
, v_return_code
, v_return_msg
);
```

© Copyright, WhereScape Software Ltd All rights reserved.

DB2

DB2 Parameters: Ws_Job_Clear_Logs_By_Date

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_job_to_clean	VARCHAR(64)	IN
p_day_count	INTEGER	IN
p_return_code	VARCHAR(1)	OUT
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT

DB2 Examples: Ws_Job_Clear_Logs_By_Date

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_sequence integer;
DECLARE p_job_name varchar(256);
DECLARE p_task_name varchar(256);
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE p_return_msg varchar(256);
DECLARE p_status integer;
DECLARE v_result_num integer;
DECLARE v_return_code varchar(1);
DECLARE v_return_msg varchar(256);
CALL [METABASE].Ws_Job_Clear_Logs_By_Date
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, 'Daily Run', 90
, v_return_code
, v_return_msg
, v_result_num
);
```

© Copyright, WhereScape Software Ltd All rights reserved.

Ws_Job_Create

Synopsis

Creates a job based on an existing job and optionally starts it immediately.

Description

Creates a job from the specified existing job, if it is in either a holding or waiting state. The new job can be started immediately. Typically, this routine is used to create & start a job from within another job. Only jobs that are in a holding or waiting state can be used as a template for the new job.

Input

Input	Description
Common Input	Includes all 5 inputs of the **Callable Routines Common Input**.
Template Job Name	The name of the job to be used as a template for the new job. The specified name must exactly match the job name as displayed by the WhereScape RED Scheduler. The specified job must be in a holding or waiting state in order to be used as a template.
New Job Name	The name of the job to be created. The new job name cannot already exist.
Job Description	A description of the new job. When not specified the setting of the Template job is copied.
New Job Status	The initial status/frequency of the new job: • **HOLD** - The status of the new job will show as 'On Hold'. The job will not run until it is subsequently released. • **ONCE** - The new job will start immediately and upon successful completion the new job will be deleted. • **ONCE+HOLD** - The new job will start immediately and upon successful completion the status will show as 'On Hold'.
Thread Count	The number of threads for the new job. When not specified the setting of the Template job is copied.
Scheduler Preference	A scheduler type or a specific scheduler name that is allowed to run the job. When not specified, the setting of the Template job is copied. **Note** Some jobs/tasks can only run in a specific environment, such as Windows or UNIX/Linux.
Maximum Logs to Retain	The maximum number of logs to retain. When not specified the setting of the Template job is copied.
Success Command	A command-line action to execute upon successful completion of the new job. When not specified, the setting of the Template job is copied. The command must be executable within the context of the scheduler that runs the job—it must be a valid Windows/UNIX/Linux command that is appropriate to the scheduler environment.
Failure Command	A command-line action to execute upon failure of the new job. When not specified the setting of the Template job is copied. The command must be executable within the context of the scheduler that runs the job—it must be a valid Windows/UNIX/Linux command that is appropriate to the scheduler environment.

Output

Output	Description
Return Code	Output Return Code: • S Success. • **N No action because the Template Job is not in a holding or waiting state.** • **P No action because the New Job name already exists.**

© Copyright, WhereScape Software Ltd All rights reserved.

Output	Description
	• E Error. • F Fatal/Unexpected Error.
Return Message	Output message indicating the action applied or the reason for no action.
Result Number	Output Result Number: • 1 Success. • **-1 Template Job is not in a holding/waiting state or the New Job name already exists.** • -2 Error. e.g. Due to invalid job name or job not running. • -3 Fatal/Unexpected Error.

SQL Server

SQL Server Parameters: Ws_Job_Create

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
@p_sequence	INTEGER	IN
@p_job_name	VARCHAR(64)	IN
@p_task_name	VARCHAR(64)	IN
@p_job_id	INTEGER	IN
@p_task_id	INTEGER	IN
@p_template_job	VARCHAR(64)	IN
@p_new_job	VARCHAR(64)	IN
@p_description	VARCHAR(256)	IN
@p_state	VARCHAR(64)	IN
@p_threads	INTEGER	IN
@p_scheduler	VARCHAR(64)	IN
@p_logs	INTEGER	IN
@p_okay	VARCHAR(256)	IN
@p_fail	VARCHAR(256)	IN
@p_att1	VARCHAR(64)	IN
@p_att2	VARCHAR(64)	IN
@p_att3	VARCHAR(64)	IN
@p_att4	VARCHAR(64)	IN
@p_att5	VARCHAR(64)	IN
@p_att6	VARCHAR(64)	IN
@p_att7	VARCHAR(64)	IN
@p_att8	VARCHAR(64)	IN
@p_return_code	VARCHAR(1)	OUT
@p_return_msg	VARCHAR(256)	OUT
@p_result	INTEGER	OUT

© Copyright, WhereScape Software Ltd All rights reserved.

SQL Server Examples: Ws_Job_Create

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE @p_sequence integer
DECLARE @p_job_name varchar(256)
DECLARE @p_task_name varchar(256)
DECLARE @p_job_id integer
DECLARE @p_task_id integer
DECLARE @p_return_msg varchar(256)
DECLARE @p_status integer
DECLARE @v_result_num integer
DECLARE @v_return_code varchar(1)
DECLARE @v_return_msg varchar(256)
EXEC Ws_Job_Create
 @p_sequence, @p_job_name, @p_task_name, @p_job_id, @p_task_id
, 'Daily Run', 'New Daily Run', 'This is the New Daily Run job.', 'ONCE'
, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL
, @v_return_code OUTPUT
, @v_return_msg OUTPUT
, @v_result_num OUTPUT
```

© Copyright, WhereScape Software Ltd All rights reserved.

Teradata

Teradata Parameters: Ws_Job_Create
Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_template_job	VARCHAR(64)	IN
p_new_job	VARCHAR(64)	IN
p_description	VARCHAR(256)	IN
p_state	VARCHAR(10)	IN
p_threads	INTEGER	IN
p_scheduler	VARCHAR(8)	IN
p_logs	INTEGER	IN
p_okay	VARCHAR(256)	IN
p_fail	VARCHAR(256)	IN
p_att1	VARCHAR(4000)	IN
p_att2	VARCHAR(4000)	IN
p_att3	VARCHAR(4000)	IN
p_att4	VARCHAR(4000)	IN
p_att5	VARCHAR(4000)	IN
p_att6	VARCHAR(4000)	IN
p_att7	VARCHAR(4000)	IN
p_att8	VARCHAR(4000)	IN
p_return_code	VARCHAR(1)	OUT
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT

Teradata Examples: Ws_Job_Create

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_sequence integer;
DECLARE p_job_name varchar(256);
DECLARE p_task_name varchar(256);
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE p_return_msg varchar(256);
DECLARE p_status integer;
DECLARE v_result_num integer;
DECLARE v_return_code varchar(1);
DECLARE v_return_msg varchar(256);
CALL [METABASE].Ws_Job_Create
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
, 'Daily Run', 'New Daily Run', 'This is the New Daily Run job.', 'ONCE'
, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL
, v_return_code
, v_return_msg
, v_result_num
);
```

Oracle

Oracle Parameters: Ws_Job_Create

Callable Routine Type: FUNCTION.

Parameter Name	Datatype	Mode
p_sequence	NUMBER	IN
p_job_name	VARCHAR2	IN
p_task_name	VARCHAR2	IN
p_job_id	NUMBER	IN
p_task_id	NUMBER	IN
p_template_job	VARCHAR2	IN
p_new_job	VARCHAR2	IN
p_description	VARCHAR2	IN
p_state	VARCHAR2	IN
p_threads	NUMBER	IN
p_scheduler	VARCHAR2	IN
p_logs	NUMBER	IN
p_okay	VARCHAR2	IN
p_fail	VARCHAR2	IN
p_att1	VARCHAR2	IN
p_att2	VARCHAR2	IN
p_att3	VARCHAR2	IN
p_att4	VARCHAR2	IN
p_att5	VARCHAR2	IN
p_att6	VARCHAR2	IN
p_att7	VARCHAR2	IN
p_att8	VARCHAR2	IN
p_return_code	VARCHAR2	OUT
p_return_msg	VARCHAR2	OUT
FUNCTION Return Value	NUMBER	OUT-Function

Oracle Examples: Ws_Job_Create

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
p_sequence number;
p_job_name varchar2(256);
p_task_name varchar2(256);
p_job_id number;
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
p_task_id number;
p_return_msg varchar2(256);
p_status number;
v_result_num number;
v_return_code varchar2(1);
v_return_msg varchar2(256);
v_result_num := Ws_Job_Create
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, 'Daily Run', 'New Daily Run', 'This is the New Daily Run job.', 'ONCE'
, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL
, v_return_code
, v_return_msg
);
```

DB2

DB2 Parameters: Ws_Job_Create
Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_template_job	VARCHAR(64)	IN
p_new_job	VARCHAR(64)	IN
p_description	VARCHAR(256)	IN
p_state	VARCHAR(10)	IN
p_threads	INTEGER	IN
p_scheduler	VARCHAR(8)	IN
p_logs	INTEGER	IN
p_okay	VARCHAR(256)	IN
p_fail	VARCHAR(256)	IN
p_att1	VARCHAR(4000)	IN
p_att2	VARCHAR(4000)	IN
p_att3	VARCHAR(4000)	IN
p_att4	VARCHAR(4000)	IN
p_att5	VARCHAR(4000)	IN
p_att6	VARCHAR(4000)	IN
p_att7	VARCHAR(4000)	IN
p_att8	VARCHAR(4000)	IN
p_return_code	VARCHAR(1)	OUT
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT

© Copyright, WhereScape Software Ltd All rights reserved.

DB2 Examples: Ws_Job_Create

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_sequence integer;
DECLARE p_job_name varchar(256);
DECLARE p_task_name varchar(256);
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE p_return_msg varchar(256);
DECLARE p_status integer;
DECLARE v_result_num integer;
DECLARE v_return_code varchar(1);
DECLARE v_return_msg varchar(256);
CALL [METABASE].Ws_Job_Create
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, 'Daily Run', 'New Daily Run', 'This is the New Daily Run job.', 'ONCE'
, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL
, v_return_code
, v_return_msg
, v_result_num
);
```

Ws_Job_CreateWait

Synopsis

Creates a job based on an existing job and schedules it to start later.

Description

Creates a job from the specified existing job, if it is in either a holding or waiting state. The new job is scheduled to start later at the specified release time. Typically, this routine is used to create & schedule a job from within another job. Only jobs that are in a holding or waiting state can be used as a template for the new job.

Input

Input	Description
Common Input	Includes all 5 inputs of the **Callable Routines Common Input**.
Template Job Name	The name of the job to be used as a template for the new job. The specified name must exactly match the job name as displayed by the WhereScape RED Scheduler. The specified job must be in a holding or waiting state in order to be used as a template.
New Job Name	The name of the job to be created. The new job name cannot already exist.
Job Description	A description of the new job. When not specified the setting of the Template job is copied.
New Job Status	The initial status/frequency of the new job: • **HOLD** - The status of the new job will show as 'On Hold'. The job will not run until it is subsequently released. • **ONCE** - The new job will start immediately and upon successful completion the new job will be deleted. • **ONCE+HOLD** - The new job will start immediately and upon successful completion the status will show as 'On Hold'.
Scheduled Release	The date/time when the new job is scheduled to be run.

© Copyright, WhereScape Software Ltd All rights reserved.

Input	Description
Date/Time	
Thread Count	The number of threads for the new job. When not specified, the setting of the Template job is copied.
Scheduler Preference	A scheduler type or a specific scheduler name that is allowed to run the job. When not specified, the setting of the Template job is copied. **Note** Some jobs/tasks can only run in a specific environment, such as Windows or UNIX/Linux.
Maximum Logs to Retain	The maximum number of logs to retain. When not specified, the setting of the Template job is copied.
Success Command	A command-line action to execute upon successful completion of the new job. When not specified, the setting of the Template job is copied. The command must be executable within the context of the scheduler that runs the job—it must be a valid Windows/UNIX/Linux command that is appropriate to the scheduler environment.
Failure Command	A command-line action to execute upon failure of the new job. When not specified the setting of the Template job is copied. The command must be executable within the context of the scheduler that runs the job—it must be a valid Windows/UNIX/Linux command that is appropriate to the scheduler environment.

Output

Output	Description
Return Code	Output Return Code: • S Success. • **N No action because the Template Job is not in a holding or waiting state.** • **P No action because the New Job name already exists.** • E Error. • F Fatal/Unexpected Error.
Return Message	Output message indicating the action applied or the reason for no action.
Result Number	Output Result Number: • 1 Success. • **-1 Template Job is not in a holding/waiting state or the New Job name already exists.** • -2 Error. e.g. Due to invalid job name or job not running. • -3 Fatal/Unexpected Error.

SQL Server

SQL Server Parameters: Ws_Job_CreateWait

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
@p_sequence	INTEGER	IN
@p_job_name	VARCHAR(64)	IN
@p_task_name	VARCHAR(64)	IN
@p_job_id	INTEGER	IN
@p_task_id	INTEGER	IN

© Copyright, WhereScape Software Ltd All rights reserved.

Parameter Name	Datatype	Mode
@p_template_job	VARCHAR(64)	IN
@p_new_job	VARCHAR(64)	IN
@p_description	VARCHAR(256)	IN
@p_state	VARCHAR(64)	IN
@p_release_time	DATETIME	IN
@p_threads	INTEGER	IN
@p_scheduler	VARCHAR(64)	IN
@p_logs	INTEGER	IN
@p_okay	VARCHAR(256)	IN
@p_fail	VARCHAR(256)	IN
@p_att1	VARCHAR(64)	IN
@p_att2	VARCHAR(64)	IN
@p_att3	VARCHAR(64)	IN
@p_att4	VARCHAR(64)	IN
@p_att5	VARCHAR(64)	IN
@p_att6	VARCHAR(64)	IN
@p_att7	VARCHAR(64)	IN
@p_att8	VARCHAR(64)	IN
@p_return_code	VARCHAR(1)	OUT
@p_return_msg	VARCHAR(256)	OUT
@p_result	INTEGER	OUT

SQL Server Examples: Ws_Job_CreateWait

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE @p_sequence integer
DECLARE @p_job_name varchar(256)
DECLARE @p_task_name varchar(256)
DECLARE @p_job_id integer
DECLARE @p_task_id integer
DECLARE @p_return_msg varchar(256)
DECLARE @p_status integer
DECLARE @v_result_num integer
DECLARE @v_return_code varchar(1)
DECLARE @v_return_msg varchar(256)
DECLARE @v_run_date datetime
SET @v_run_date = DATEADD(MONTH, 1, GETDATE())
EXEC Ws_Job_CreateWait
 @p_sequence, @p_job_name, @p_task_name, @p_job_id, @p_task_id
, 'Daily Run', 'New Daily Run2', 'This is the New Daily Run job.', 'ONCE'
, @v_run_date
, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL
, @v_return_code OUTPUT
, @v_return_msg OUTPUT
, @v_result_num OUTPUT
```

© Copyright, WhereScape Software Ltd All rights reserved.

Teradata

Teradata Parameters: Ws_Job_CreateWait

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_template_job	VARCHAR(64)	IN
p_new_job	VARCHAR(64)	IN
p_description	VARCHAR(256)	IN
p_state	VARCHAR(10)	IN
p_release_time	TIMESTAMP	IN
p_threads	INTEGER	IN
p_scheduler	VARCHAR(8)	IN
p_logs	INTEGER	IN
p_okay	VARCHAR(256)	IN
p_fail	VARCHAR(256)	IN
p_att1	VARCHAR(4000)	IN
p_att2	VARCHAR(4000)	IN
p_att3	VARCHAR(4000)	IN
p_att4	VARCHAR(4000)	IN
p_att5	VARCHAR(4000)	IN
p_att6	VARCHAR(4000)	IN
p_att7	VARCHAR(4000)	IN
p_att8	VARCHAR(4000)	IN
p_return_code	VARCHAR(1)	OUT
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT

Teradata Examples: Ws_Job_CreateWait

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_sequence integer;
DECLARE p_job_name varchar(256);
DECLARE p_task_name varchar(256);
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE p_return_msg varchar(256);
DECLARE p_status integer;
DECLARE v_result_num integer;
DECLARE v_return_code varchar(1);
DECLARE v_return_msg varchar(256);
CALL [METABASE].Ws_Job_CreateWait
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, 'Daily Run', 'New Daily Run', 'This is the New Daily Run job.', 'ONCE'
, (CURRENT_TIMESTAMP + INTERVAL '1' MONTH)
, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL
, v_return_code
, v_return_msg
, v_result_num
);
```

Oracle

Oracle Parameters: Ws_Job_CreateWait

Callable Routine Type: FUNCTION.

Parameter Name	Datatype	Mode
p_sequence	NUMBER	IN
p_job_name	VARCHAR2	IN
p_task_name	VARCHAR2	IN
p_job_id	NUMBER	IN
p_task_id	NUMBER	IN
p_template_job	VARCHAR2	IN
p_new_job	VARCHAR2	IN
p_description	VARCHAR2	IN
p_state	VARCHAR2	IN
p_release_time	DATE	IN
p_threads	NUMBER	IN
p_scheduler	VARCHAR2	IN
p_logs	NUMBER	IN
p_okay	VARCHAR2	IN
p_fail	VARCHAR2	IN
p_att1	VARCHAR2	IN
p_att2	VARCHAR2	IN
p_att3	VARCHAR2	IN
p_att4	VARCHAR2	IN
p_att5	VARCHAR2	IN
p_att6	VARCHAR2	IN
p_att7	VARCHAR2	IN
p_att8	VARCHAR2	IN
p_return_code	VARCHAR2	OUT
p_return_msg	VARCHAR2	OUT
FUNCTION Return Value	NUMBER	OUT-Function

© Copyright, WhereScape Software Ltd All rights reserved.

Oracle Examples: Ws_Job_CreateWait

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
p_sequence number;
p_job_name varchar2(256);
p_task_name varchar2(256);
p_job_id number;
p_task_id number;
p_return_msg varchar2(256);
p_status number;
v_result_num number;
v_return_code varchar2(1);
v_return_msg varchar2(256);
v_result_num := Ws_Job_CreateWait
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, 'Daily Run', 'New Daily Run', 'This is the New Daily Run job.', 'ONCE'
, ADD_MONTHS(SYSDATE, 1)
, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL
, v_return_code
, v_return_msg
);
```

DB2

DB2 Parameters: Ws_Job_CreateWait
Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_template_job	VARCHAR(64)	IN
p_new_job	VARCHAR(64)	IN
p_description	VARCHAR(256)	IN
p_state	VARCHAR(10)	IN
p_release_time	TIMESTAMP	IN
p_threads	INTEGER	IN
p_scheduler	VARCHAR(8)	IN
p_logs	INTEGER	IN
p_okay	VARCHAR(256)	IN
p_fail	VARCHAR(256)	IN
p_att1	VARCHAR(4000)	IN
p_att2	VARCHAR(4000)	IN
p_att3	VARCHAR(4000)	IN
p_att4	VARCHAR(4000)	IN
p_att5	VARCHAR(4000)	IN

© Copyright, WhereScape Software Ltd All rights reserved.

Parameter Name	Datatype	Mode
p_att6	VARCHAR(4000)	IN
p_att7	VARCHAR(4000)	IN
p_att8	VARCHAR(4000)	IN
p_return_code	VARCHAR(1)	OUT
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT

DB2 Examples: Ws_Job_CreateWait

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_sequence integer;
DECLARE p_job_name varchar(256);
DECLARE p_task_name varchar(256);
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE p_return_msg varchar(256);
DECLARE p_status integer;
DECLARE v_result_num integer;
DECLARE v_return_code varchar(1);
DECLARE v_return_msg varchar(256);
CALL [METABASE].Ws_Job_CreateWait
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, 'Daily Run', 'New Daily Run', 'This is the New Daily Run job.', 'ONCE'
, (CURRENT TIMESTAMP + 1 MONTH)
, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL
, v_return_code
, v_return_msg
, v_result_num
);
```

Ws_Job_Dependency

Synopsis

Adds or removes a child-to-parent dependency between two jobs to control the child job.

Description

Adds or removes a child-to-parent dependency between two jobs to control the child job. The dependent child job can be defined to fail (if necessary) when the parent job does not complete successfully in the required timeframe. The acceptable timeframe can be defined in terms of the maximum minutes in the past, to look back and the maximum minutes in the future to wait for successful completion of the parent job.

Input

Input	Description
Common Input	Includes all 5 inputs of the **Callable Routines Common Input**.
Action	Either '**ADD**' or '**DELETE**' the job dependency.
Parent Job Name	The name of Parent Job that the Child Job will depend on.
Child Job Name	The name of the Child Job that will be dependent on the Parent Job.
Parent Success Required	Indicates whether or not the Child Job will fail when the Parent Job does not complete successfully in the required time frame.

© Copyright, WhereScape Software Ltd All rights reserved.

Input	Description
Maximum Look Back Minutes	The Maximum minutes in the past to look back for successful completion of the Parent Job.
Maximum Wait Minutes	The Maximum minutes in the future to wait for successful completion of the Parent Job.

Output

Output	Description
Return Code	Output Return Code: • S Success. • **W Warning. Dependency already exists ('ADD' action) or does not exist ('DELETE' action).** • E Error. • F Fatal/Unexpected Error.
Return Message	Output message indicating the action applied or the reason for no action.
Result Number	Output Result Number: • 1 Success. • **-1 Warning. Dependency already exists ('ADD' action) or does not exist ('DELETE' action).** • -2 Error. e.g. Due to invalid job name or job not running. • -3 Fatal/Unexpected Error.

SQL Server

SQL Server Parameters: Ws_Job_Dependency

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
@p_sequence	INTEGER	IN
@p_job_name	VARCHAR(64)	IN
@p_task_name	VARCHAR(64)	IN
@p_job_id	INTEGER	IN
@p_task_id	INTEGER	IN
@p_action	VARCHAR(64)	IN
@p_parent	VARCHAR(64)	IN
@p_child	VARCHAR(64)	IN
@p_required	VARCHAR(1)	IN
@p_look_back	INTEGER	IN
@p_max_wait	INTEGER	IN
@p_return_code	VARCHAR(1)	OUT
@p_return_msg	VARCHAR(256)	OUT
@p_result	INTEGER	OUT

SQL Server Examples: Ws_Job_Dependency

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE @p_sequence integer
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
DECLARE @p_job_name varchar(256)
DECLARE @p_task_name varchar(256)
DECLARE @p_job_id integer
DECLARE @p_task_id integer
DECLARE @p_return_msg varchar(256)
DECLARE @p_status integer
DECLARE @v_result_num integer
DECLARE @v_return_code varchar(1)
DECLARE @v_return_msg varchar(256)
EXEC Ws_Job_Dependency
 @p_sequence, @p_job_name, @p_task_name, @p_job_id, @p_task_id
, 'ADD', 'Daily Run', 'Daily Run Part2', 'Y', 60, 60
, @v_return_code OUTPUT
, @v_return_msg OUTPUT
, @v_result_num OUTPUT
```

Teradata

Teradata Parameters: Ws_Job_Dependency
Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_action	VARCHAR(10)	IN
p_parent	VARCHAR(64)	IN
p_child	VARCHAR(64)	IN
p_required	VARCHAR(1)	IN
p_look_back	INTEGER	IN
p_max_wait	INTEGER	IN
p_return_code	VARCHAR(1)	OUT
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT

Teradata Examples: Ws_Job_Dependency

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_sequence integer;
DECLARE p_job_name varchar(256);
DECLARE p_task_name varchar(256);
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE p_return_msg varchar(256);
DECLARE p_status integer;
DECLARE v_result_num integer;
DECLARE v_return_code varchar(1);
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
DECLARE v_return_msg varchar(256);
CALL [METABASE].Ws_Job_Dependency
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, 'ADD', 'Daily Run', 'Daily Run Part2', 'Y', 60, 60
, v_return_code
, v_return_msg
, v_result_num
);
```

Oracle

Oracle Parameters: Ws_Job_Dependency

Callable Routine Type: FUNCTION.

Parameter Name	Datatype	Mode
p_sequence	NUMBER	IN
p_job_name	VARCHAR2	IN
p_task_name	VARCHAR2	IN
p_job_id	NUMBER	IN
p_task_id	NUMBER	IN
p_action	VARCHAR2	IN
p_parent	VARCHAR2	IN
p_child	VARCHAR2	IN
p_required	VARCHAR2	IN
p_look_back	NUMBER	IN
p_max_wait	NUMBER	IN
p_return_code	VARCHAR2	OUT
p_return_msg	VARCHAR2	OUT
FUNCTION Return Value	NUMBER	OUT-Function

Oracle Examples: Ws_Job_Dependency

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
p_sequence number;
p_job_name varchar2(256);
p_task_name varchar2(256);
p_job_id number;
p_task_id number;
p_return_msg varchar2(256);
p_status number;
v_result_num number;
v_return_code varchar2(1);
v_return_msg varchar2(256);
v_result_num := Ws_Job_Dependency
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, 'ADD', 'Daily Run', 'Daily Run Part2', 'Y', 60, 60
, v_return_code
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
, v_return_msg
);
```

DB2

DB2 Parameters: Ws_Job_Dependency

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_action	VARCHAR(10)	IN
p_parent	VARCHAR(64)	IN
p_child	VARCHAR(64)	IN
p_required	VARCHAR(1)	IN
p_look_back	INTEGER	IN
p_max_wait	INTEGER	IN
p_return_code	VARCHAR(1)	OUT
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT

DB2 Examples: Ws_Job_Dependency

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_sequence integer;
DECLARE p_job_name varchar(256);
DECLARE p_task_name varchar(256);
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE p_return_msg varchar(256);
DECLARE p_status integer;
DECLARE v_result_num integer;
DECLARE v_return_code varchar(1);
DECLARE v_return_msg varchar(256);
CALL [METABASE].Ws_Job_Dependency
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, 'ADD', 'Daily Run', 'Daily Run Part2', 'Y', 60, 60
, v_return_code
, v_return_msg
, v_result_num
);
```

© Copyright, WhereScape Software Ltd All rights reserved.

Ws_Job_Release

Synopsis

Starts a job if it is in a holding or waiting state.

Description

Releases the specified job, if it is in a holding or waiting state, which sets the start time to the current time so that it starts immediately. Typically, this routine is used to start a job from within another job or via a third-party scheduler (rather than a WhereScape RED Scheduler).

Input

Input	Description
Common Input	Includes all 5 inputs of the **Callable Routines Common Input**.
Release Job Name	The name of the job to be started/released. The specified name must exactly match the job name as displayed by the WhereScape RED Scheduler. The specified job must be in a holding or waiting state in order to be released.

Output

Output	Description
Return Code	Output Return Code: • S Success. • **N No action because the Template Job is not in a holding or waiting state**. • E Error. • F Fatal/Unexpected Error.
Return Message	Output message indicating the action applied or the reason for no action.
Result Number	Output Result Number: • 1 Success. • **-1 No action because the job is not in a holding or waiting state**. • -2 Error. • -3 Fatal/Unexpected Error.

SQL Server

SQL Server Parameters: Ws_Job_Release

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
@p_sequence	INTEGER	IN
@p_job_name	VARCHAR(64)	IN
@p_task_name	VARCHAR(64)	IN
@p_job_id	INTEGER	IN
@p_task_id	INTEGER	IN
@p_release_job	VARCHAR(64)	IN
@p_return_code	VARCHAR(1)	OUT
@p_return_msg	VARCHAR(256)	OUT
@p_result	INTEGER	OUT

SQL Server Examples: Ws_Job_Release

© Copyright, WhereScape Software Ltd All rights reserved.

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE @p_sequence integer
DECLARE @p_job_name varchar(256)
DECLARE @p_task_name varchar(256)
DECLARE @p_job_id integer
DECLARE @p_task_id integer
DECLARE @p_return_msg varchar(256)
DECLARE @p_status integer
DECLARE @v_result_num integer
DECLARE @v_return_code varchar(1)
DECLARE @v_return_msg varchar(256)
EXEC Ws_Job_Release
 @p_sequence, @p_job_name, @p_task_name, @p_job_id, @p_task_id
, 'Daily Run'
, @v_return_code OUTPUT
, @v_return_msg OUTPUT
, @v_result_num OUTPUT
```

Teradata

Teradata Parameters: Ws_Job_Release
Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_release_job	VARCHAR(64)	IN
p_return_code	VARCHAR(1)	OUT
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT

Teradata Examples: Ws_Job_Release

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_sequence integer;
DECLARE p_job_name varchar(256);
DECLARE p_task_name varchar(256);
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE p_return_msg varchar(256);
DECLARE p_status integer;
DECLARE v_result_num integer;
DECLARE v_return_code varchar(1);
DECLARE v_return_msg varchar(256);
CALL [METABASE].Ws_Job_Release
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
, 'Daily Run'
, v_return_code
, v_return_msg
, v_result_num
);
```

Oracle

Oracle Parameters: Ws_Job_Release

Callable Routine Type: FUNCTION.

Parameter Name	Datatype	Mode
p_sequence	NUMBER	IN
p_job_name	VARCHAR2	IN
p_task_name	VARCHAR2	IN
p_job_id	NUMBER	IN
p_task_id	NUMBER	IN
p_release_job	VARCHAR2	IN
p_return_code	VARCHAR2	OUT
p_return_msg	VARCHAR2	OUT
FUNCTION Return Value	NUMBER	OUT-Function

Oracle Examples: Ws_Job_Release

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
p_sequence number;
p_job_name varchar2(256);
p_task_name varchar2(256);
p_job_id number;
p_task_id number;
p_return_msg varchar2(256);
p_status number;
v_result_num number;
v_return_code varchar2(1);
v_return_msg varchar2(256);
v_result_num := Ws_Job_Release
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, 'Daily Run'
, v_return_code
, v_return_msg
);
```

© Copyright, WhereScape Software Ltd All rights reserved.

DB2

DB2 Parameters: Ws_Job_Release

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_release_job	VARCHAR(64)	IN
p_return_code	VARCHAR(1)	OUT
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT

DB2 Examples: Ws_Job_Release

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_sequence integer;
DECLARE p_job_name varchar(256);
DECLARE p_task_name varchar(256);
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE p_return_msg varchar(256);
DECLARE p_status integer;
DECLARE v_result_num integer;
DECLARE v_return_code varchar(1);
DECLARE v_return_msg varchar(256);
CALL [METABASE].Ws_Job_Release
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, 'Daily Run'
, v_return_code
, v_return_msg
, v_result_num
);
```

Ws_Job_Restart

Synopsis

Starts a job if it is in a failed state.

Description

Releases the specified job if it is in a failed state, which sets the start time to the current time so that it starts immediately. This routine can be executed as part of a database start-up sequence to restart each failed job that may have stopped, due to an earlier database shutdown.

Input

Input	Description
Common Input	Includes all 5 inputs of the **Callable Routines Common Input**.
Restart Job Name	The name of the job to be restarted/released. The specified name must exactly match

© Copyright, WhereScape Software Ltd All rights reserved.

Input	Description
	the job name as displayed by the WhereScape RED Scheduler. The specified job must be in a failed state in order to be restarted.

Output

Output	Description
Return Code	Output Return Code: • S Success. • **N No action because the job is not in a failed state.** • **R No action because the job is currently running.** • **U No action because the job is in an unusual state due to an error (result number -2). The job is classified as running but it is NOT actually running or failed so it cannot be restarted. This may occur if the scheduler has failed and the job is in a pending state.** • E Error. • F Fatal/Unexpected Error.
Return Message	Output message indicating the action applied or the reason for no action.
Result Number	Output Result Number: • 1 Success. • **-1 No action because the job is not in a failed state or it is currently running.** • -2 Error. • -3 Fatal/Unexpected Error.

SQL Server

SQL Server Parameters: Ws_Job_Restart

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
@p_sequence	INTEGER	IN
@p_job_name	VARCHAR(64)	IN
@p_task_name	VARCHAR(64)	IN
@p_job_id	INTEGER	IN
@p_task_id	INTEGER	IN
@p_restart_job	VARCHAR(64)	IN
@p_return_code	VARCHAR(1)	OUT
@p_return_msg	VARCHAR(256)	OUT
@p_result	INTEGER	OUT

SQL Server Examples: Ws_Job_Restart

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE @p_sequence integer
DECLARE @p_job_name varchar(256)
DECLARE @p_task_name varchar(256)
DECLARE @p_job_id integer
DECLARE @p_task_id integer
DECLARE @p_return_msg varchar(256)
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
DECLARE @p_status integer
DECLARE @v_result_num integer
DECLARE @v_return_code varchar(1)
DECLARE @v_return_msg varchar(256)
EXEC Ws_Job_Restart
 @p_sequence, @p_job_name, @p_task_name, @p_job_id, @p_task_id
, 'Daily Run'
, @v_return_code OUTPUT
, @v_return_msg OUTPUT
, @v_result_num OUTPUT
```

Teradata

Teradata Parameters: Ws_Job_Restart

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_restart_job	VARCHAR(64)	IN
p_return_code	VARCHAR(1)	OUT
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT

Teradata Examples: Ws_Job_Restart

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_sequence integer;
DECLARE p_job_name varchar(256);
DECLARE p_task_name varchar(256);
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE p_return_msg varchar(256);
DECLARE p_status integer;
DECLARE v_result_num integer;
DECLARE v_return_code varchar(1);
DECLARE v_return_msg varchar(256);
CALL [METABASE].Ws_Job_Restart
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, 'Daily Run'
, v_return_code
, v_return_msg
, v_result_num
);
```

© Copyright, WhereScape Software Ltd All rights reserved.

Oracle

Oracle Parameters: Ws_Job_Restart

Callable Routine Type: FUNCTION.

Parameter Name	Datatype	Mode
p_sequence	NUMBER	IN
p_job_name	VARCHAR2	IN
p_task_name	VARCHAR2	IN
p_job_id	NUMBER	IN
p_task_id	NUMBER	IN
p_restart_job	VARCHAR2	IN
p_return_code	VARCHAR2	OUT
p_return_msg	VARCHAR2	OUT
FUNCTION Return Value	NUMBER	OUT-Function

Oracle Examples: Ws_Job_Restart

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
p_sequence number;
p_job_name varchar2(256);
p_task_name varchar2(256);
p_job_id number;
p_task_id number;
p_return_msg varchar2(256);
p_status number;
v_result_num number;
v_return_code varchar2(1);
v_return_msg varchar2(256);
v_result_num := Ws_Job_Restart
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, 'Daily Run'
, v_return_code
, v_return_msg
);
```

DB2

DB2 Parameters: Ws_Job_Restart

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_restart_job	VARCHAR(64)	IN
p_return_code	VARCHAR(1)	OUT

© Copyright, WhereScape Software Ltd All rights reserved.

Parameter Name	Datatype	Mode
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT

DB2 Examples: Ws_Job_Restart

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_sequence integer;
DECLARE p_job_name varchar(256);
DECLARE p_task_name varchar(256);
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE p_return_msg varchar(256);
DECLARE p_status integer;
DECLARE v_result_num integer;
DECLARE v_return_code varchar(1);
DECLARE v_return_msg varchar(256);
CALL [METABASE].Ws_Job_Restart
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, 'Daily Run'
, v_return_code
, v_return_msg
, v_result_num
);
```

Ws_Job_Schedule

Synopsis

Schedules a job if it is in a holding or waiting state.

Description

Schedules the specified job if it is in a holding or waiting state, which will start at the specified time. Typically, this routine is used to schedule a job from within another job.

Input

Input	Description
Common Input	Includes all 5 inputs of the **Callable Routines Common Input**.
Schedule Job Name	The name of the job to be scheduled. The specified name must exactly match the job name as displayed by the WhereScape RED Scheduler. The specified job must be in a holding or waiting state in order to be scheduled.
Scheduled Release Time	The date/time that the job is to be scheduled to be released/started.

Output

Output	Description
Return Code	Output Return Code: • S Success. • **N No action because the job is not in a holding or waiting state.** • E Error. • F Fatal/Unexpected Error.
Return Message	Output message indicating the action applied or the reason for no action.

© Copyright, WhereScape Software Ltd All rights reserved.

Output	Description
Result Number	Output Result Number: • 1 Success. • **-1 No action because the job is not in a holding or waiting state.** • -2 Error. • -3 Fatal/Unexpected Error.

SQL Server

SQL Server Parameters: Ws_Job_Schedule

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
@p_sequence	INTEGER	IN
@p_job_name	VARCHAR(64)	IN
@p_task_name	VARCHAR(64)	IN
@p_job_id	INTEGER	IN
@p_task_id	INTEGER	IN
@p_release_job	VARCHAR(64)	IN
@p_release_time	DATETIME	IN
@p_return_code	VARCHAR(1)	OUT
@p_return_msg	VARCHAR(256)	OUT
@p_result	INTEGER	OUT

SQL Server Examples: Ws_Job_Schedule

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE @p_sequence integer
DECLARE @p_job_name varchar(256)
DECLARE @p_task_name varchar(256)
DECLARE @p_job_id integer
DECLARE @p_task_id integer
DECLARE @p_return_msg varchar(256)
DECLARE @p_status integer
DECLARE @v_result_num integer
DECLARE @v_return_code varchar(1)
DECLARE @v_return_msg varchar(256)
DECLARE @v_run_date datetime
SET @v_run_date = GETDATE() + 1
EXEC Ws_Job_Schedule
 @p_sequence, @p_job_name, @p_task_name, @p_job_id, @p_task_id
, 'Daily Run', @v_run_date
, @v_return_code OUTPUT
, @v_return_msg OUTPUT
, @v_result_num OUTPUT
```

© Copyright, WhereScape Software Ltd All rights reserved.

Teradata

Teradata Parameters: Ws_Job_Schedule
Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_release_job	VARCHAR(64)	IN
p_release_time	TIMESTAMP	IN
p_return_code	VARCHAR(1)	OUT
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT

Teradata Examples: Ws_Job_Schedule

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_sequence integer;
DECLARE p_job_name varchar(256);
DECLARE p_task_name varchar(256);
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE p_return_msg varchar(256);
DECLARE p_status integer;
DECLARE v_result_num integer;
DECLARE v_return_code varchar(1);
DECLARE v_return_msg varchar(256);
CALL [METABASE].Ws_Job_Schedule
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, 'Daily Run', CURRENT_TIMESTAMP + INTERVAL '1' DAY
, v_return_code
, v_return_msg
, v_result_num
);
```

Oracle

Oracle Parameters: Ws_Job_Schedule
Callable Routine Type: FUNCTION.

Parameter Name	Datatype	Mode
p_sequence	NUMBER	IN
p_job_name	VARCHAR2	IN
p_task_name	VARCHAR2	IN
p_job_id	NUMBER	IN

© Copyright, WhereScape Software Ltd All rights reserved.

Parameter Name	Datatype	Mode
p_task_id	NUMBER	IN
p_release_job	VARCHAR2	IN
p_release_time	DATE	IN
p_return_code	VARCHAR2	OUT
p_return_msg	VARCHAR2	OUT
FUNCTION Return Value	NUMBER	OUT-Function

Oracle Examples: Ws_Job_Schedule

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
p_sequence number;
p_job_name varchar2(256);
p_task_name varchar2(256);
p_job_id number;
p_task_id number;
p_return_msg varchar2(256);
p_status number;
v_result_num number;
v_return_code varchar2(1);
v_return_msg varchar2(256);
v_result_num := Ws_Job_Schedule
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, 'Daily Run', SYSDATE+1
, v_return_code
, v_return_msg
);
```

DB2

DB2 Parameters: Ws_Job_Schedule

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_release_job	VARCHAR(64)	IN
p_release_time	TIMESTAMP	IN
p_return_code	VARCHAR(1)	OUT
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT

DB2 Examples: Ws_Job_Schedule

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_sequence integer;
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
DECLARE p_job_name varchar(256);
DECLARE p_task_name varchar(256);
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE p_return_msg varchar(256);
DECLARE p_status integer;
DECLARE v_result_num integer;
DECLARE v_return_code varchar(1);
DECLARE v_return_msg varchar(256);
CALL [METABASE].Ws_Job_Schedule
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, 'Daily Run', CURRENT TIMESTAMP + 1 DAY
, v_return_code
, v_return_msg
, v_result_num
);
```

Ws_Job_Status

Synopsis

Returns the current status of a job.

Description

Returns the current status of the specified job, as recorded by a WhereScape RED Scheduler. Typically, this routine is used by a third-party scheduler or a user-defined procedure/script to check on a job.

Input

Input	Description
Common Input	Includes all 5 inputs of the **Callable Routines Common Input**.
Job Sequence	The unique integer identifier of the job to return the status of. This input is optional but when it is specified, the started **within** and started **after** inputs should not be specified.
Job Name	The name of the job to return the status of. The specified name must exactly match the job name as displayed by the WhereScape RED Scheduler.
Started Within Last Minutes	The maximum minutes [0-148599] (up to ~103.1 days) in the past to look back for the job to have started. This input is optional but when it is specified, the job name must be specified and the job sequence and started **after** inputs should not be specified. Note: If multiple instances of the job have started in the specified time frame then the last job to start is returned (i.e. the job with the highest sequence number).
Started After Time	The date/time after which to look for the job to have started. This input is optional but when it is specified, the job name must be specified and the job sequence and started **within** inputs should not be specified. **Note** If multiple instances of the job have started in the specified time frame then the last job to start is returned (i.e. the job with the highest sequence number).

Output

Output	Description
Return Code	Output Return Code: • S - **Success**. • N - **The job exists but it was NOT started within the specified time frame.** • E - **Error**.

© Copyright, WhereScape Software Ltd All rights reserved.

Output	Description
	• F - **Fatal/Unexpected Error**.
Return Message	Output message indicating the action applied or the reason for no action.
Result Number	Output Result Number: • 1 - **Success**. • -1 - **The job exists but it was NOT started within the specified time frame.** • -2 - **Error**. • -3 -**Fatal/Unexpected Error**. • 0 - see note below.
Simplified Job Status Code	Simplified Job Status Code: • N - **Not Running**. • R - **Running**. • F - **Failed**. • C - **Completed**. • 0 - see note below.
Standard Job Status Code	Standard Job Status Code: • H - **On Hold**. The job is on hold. A held job can be edited and/or started. • W - **Waiting**. The job is waiting to start (it is either waiting for the scheduled time to arrive or is waiting for an available scheduler). • B - **Blocked**. The job is blocked because a previous instance of the same job is still running. • P - **Pending**. This is the initial interim status of an "about to start running" job. The scheduler has identified that the job is ready to start and is preparing to run it. A job should only be pending for a brief period so if it remains pending for a prolonged period then an unexpected error has occurred. • R - **Running**. The job is currently running. • F - **Failed**. The job failed due to an error. • C - **Completed**. The job completed successfully (but it may have warnings). A completed job cannot be restarted. • G - **Failed - Aborted**. The job failed and was subsequently aborted. An aborted job cannot be restarted. • E - **Error Completion**. • 0 - see note below.
Enhanced Job Status Number	Enhanced Job Status Number that returns an **integer** rather than the standard alphabetic code. The **running** and **completed statuses are enhanced** to distinguish errors or warnings. 1. **On Hold.** The job is on hold. A held job can be edited and/or started. 2. **Waiting**. The job is waiting to start (it is either waiting for the scheduled time to arrive or is waiting for an available scheduler). 3. **Blocked**. The job is blocked because a previous instance of the same job is still running. 4. **Pending**. This is the initial interim status of an "about to start running" job. The scheduler has identified that the job is ready to start and is preparing to run it. A job should only be pending for a brief period so if it remains pending for a prolonged period then an unexpected error has occurred. 5. **Running**. The job is currently running and no tasks have failed or produced warnings. 6. **Running with Errors**. The job is currently running but some tasks have failed. The job will ultimately fail when all the tasks that are NOT dependent on the failed tasks have finished. 7. **Running with Warnings**. The job is currently running and some tasks have produced warnings.

© Copyright, WhereScape Software Ltd All rights reserved.

Output	Description
	8. **Failed**. The job failed due to an error. 9. **Completed**. The job completed without warnings. A completed job cannot be restarted. 10. **Completed with Warnings**. The job completed with warnings. A completed job cannot be restarted. 11. **Failed - Aborted**. The job failed and it was subsequently aborted. An aborted job cannot be restarted. 12. **Error Completion**. 0 - see note below.

Notes

All three returned status values can also return '**0**' in any of the following situations:

- Illegal combination of parameters specified.
- Unable to locate specified job sequence.
- Unable to locate specified job name.
- Job Not Found having started in the last **SpecifiedMinutes** minutes.
- Job Not Found having started after **SpecifiedDateTime**.

SQL Server

SQL Server Parameters: Ws_Job_Status

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
@p_sequence	INTEGER	IN
@p_job_name	VARCHAR(64)	IN
@p_task_name	VARCHAR(64)	IN
@p_job_id	INTEGER	IN
@p_task_id	INTEGER	IN
@p_check_sequence	INTEGER	IN
@p_check_job	VARCHAR(64)	IN
@p_started_in_last_mi	INTEGER	IN
@p_started_after_dt	DATETIME	IN
@p_return_code	VARCHAR(1)	OUT
@p_return_msg	VARCHAR(256)	OUT
@p_result	INTEGER	OUT
@p_job_status_simple	VARCHAR(1)	OUT
@p_job_status_standard	VARCHAR(1)	OUT
@p_job_status_enhanced	VARCHAR(2)	OUT

SQL Server Examples: Ws_Job_Status

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE @p_sequence integer
DECLARE @p_job_name varchar(256)
DECLARE @p_task_name varchar(256)
DECLARE @p_job_id integer
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
DECLARE @p_task_id integer
DECLARE @p_return_msg varchar(256)
DECLARE @p_status integer
DECLARE @v_result_num integer
DECLARE @v_return_code varchar(1)
DECLARE @v_return_msg varchar(256)
DECLARE @v_job_status_simple varchar(1)
DECLARE @v_job_status_standard varchar(1)
DECLARE @v_job_status_enhanced varchar(2)
EXEC Ws_Job_Status
 @p_sequence, @p_job_name, @p_task_name, @p_job_id, @p_task_id
, NULL, 'Daily Run', 10, NULL
, @v_return_code OUTPUT
, @v_return_msg OUTPUT
, @v_result_num OUTPUT
, @v_job_status_simple OUTPUT
, @v_job_status_standard OUTPUT
, @v_job_status_enhanced OUTPUT
```

Teradata

Teradata Parameters: Ws_Job_Status
Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_check_sequence	INTEGER	IN
p_check_job	VARCHAR(64)	IN
p_started_in_last_mi	INTEGER	IN
p_started_after_dt	TIMESTAMP	IN
p_return_code	VARCHAR(1)	OUT
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT
p_job_status_simple	VARCHAR(1)	OUT
p_job_status_standard	VARCHAR(1)	OUT
p_job_status_enhanced	VARCHAR(2)	OUT

Teradata Examples: Ws_Job_Status

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_sequence integer;
DECLARE p_job_name varchar(256);
DECLARE p_task_name varchar(256);
DECLARE p_job_id integer;
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
DECLARE p_task_id integer;
DECLARE p_return_msg varchar(256);
DECLARE p_status integer;
DECLARE v_result_num integer;
DECLARE v_return_code varchar(1);
DECLARE v_return_msg varchar(256);
DECLARE v_job_status_simple varchar(1);
DECLARE v_job_status_standard varchar(1);
DECLARE v_job_status_enhanced varchar(2);
CALL [METABASE].Ws_Job_Status
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, NULL, 'Daily Run', 10, NULL
, v_return_code
, v_return_msg
, v_result_num
, v_job_status_simple, v_job_status_standard, v_job_status_enhanced
);
```

Oracle

Oracle Parameters: Ws_Job_Status

Callable Routine Type: FUNCTION.

Parameter Name	Datatype	Mode
p_sequence	NUMBER	IN
p_job_name	VARCHAR2	IN
p_task_name	VARCHAR2	IN
p_job_id	NUMBER	IN
p_task_id	NUMBER	IN
p_check_sequence	NUMBER	IN
p_check_job	VARCHAR2	IN
p_started_in_last_mi	NUMBER	IN
p_started_after_dt	DATE	IN
p_return_code	VARCHAR2	OUT
p_return_msg	VARCHAR2	OUT
p_job_status_simple	VARCHAR2	OUT
p_job_status_standard	VARCHAR2	OUT
p_job_status_enhanced	VARCHAR2	OUT
FUNCTION Return Value	NUMBER	OUT-Function

Oracle Examples: Ws_Job_Status

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
p_sequence number;
p_job_name varchar2(256);
p_task_name varchar2(256);
p_job_id number;
p_task_id number;
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
p_return_msg varchar2(256);
p_status number;
v_result_num number;
v_return_code varchar2(1);
v_return_msg varchar2(256);
v_job_status_simple varchar2(1);
v_job_status_standard varchar2(1);
v_job_status_enhanced varchar2(2);
v_result_num := Ws_Job_Status
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, NULL, 'Daily Run', 10, NULL
, v_return_code
, v_return_msg
, v_job_status_simple, v_job_status_standard, v_job_status_enhanced
);
```

DB2

DB2 Parameters: Ws_Job_Status

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_check_sequence	INTEGER	IN
p_check_job	VARCHAR(64)	IN
p_started_in_last_mi	INTEGER	IN
p_started_after_dt	TIMESTAMP	IN
p_return_code	VARCHAR(1)	OUT
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT
p_job_status_simple	VARCHAR(1)	OUT
p_job_status_standard	VARCHAR(1)	OUT
p_job_status_enhanced	VARCHAR(2)	OUT

DB2 Examples: Ws_Job_Status

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_sequence integer;
DECLARE p_job_name varchar(256);
DECLARE p_task_name varchar(256);
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE p_return_msg varchar(256);
DECLARE p_status integer;
DECLARE v_result_num integer;
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
DECLARE v_return_code varchar(1);
DECLARE v_return_msg varchar(256);
DECLARE v_job_status_simple varchar(1);
DECLARE v_job_status_standard varchar(1);
DECLARE v_job_status_enhanced varchar(2);
CALL [METABASE].Ws_Job_Status
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, NULL, 'Daily Run', 10, NULL
, v_return_code
, v_return_msg
, v_result_num
, v_job_status_simple, v_job_status_standard, v_job_status_enhanced
);
```

Ws_Load_Change

Synopsis

Changes the Connection or Schema of a load table.

Description

Changes either the Connection or the Schema of the specified load table. Only the Connection or Schema can be changed so two calls are required to change both.

Input

Input	Description
Common Input	Includes all 5 inputs of the **Callable Routines Common Input**.
Change Property	Change either the '**SCHEMA**' or the '**CONNECTION**' of the specified load table. Separate calls must be made if both the schema and connection need to be changed.
Load Table Name	The name of the load table to be changed.
New Property Value	Either the new schema name or the new connection name.

Output

Output	Description
Return Code	Output Return Code: • S Success. • E Error. • F Fatal/Unexpected Error.
Return Message	Output message indicating the action applied or the reason for no action.
Result Number	Output Result Number: • 1 Success. • -2 Error. • -3 Fatal/Unexpected Error.

© Copyright, WhereScape Software Ltd All rights reserved.

SQL Server

SQL Server Parameters: Ws_Load_Change

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
@p_sequence	INTEGER	IN
@p_job_name	VARCHAR(64)	IN
@p_task_name	VARCHAR(64)	IN
@p_job_id	INTEGER	IN
@p_task_id	INTEGER	IN
@p_action	VARCHAR(64)	IN
@p_table	VARCHAR(64)	IN
@p_new_value	VARCHAR(64)	IN
@p_return_code	VARCHAR(1)	OUT
@p_return_msg	VARCHAR(256)	OUT
@p_result	INTEGER	OUT

SQL Server Examples: Ws_Load_Change

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE @p_sequence integer
DECLARE @p_job_name varchar(256)
DECLARE @p_task_name varchar(256)
DECLARE @p_job_id integer
DECLARE @p_task_id integer
DECLARE @p_return_msg varchar(256)
DECLARE @p_status integer
DECLARE @v_result_num integer
DECLARE @v_return_code varchar(1)
DECLARE @v_return_msg varchar(256)
EXEC Ws_Load_Change
 @p_sequence, @p_job_name, @p_task_name, @p_job_id, @p_task_id
, 'CONNECTION', 'load_customer', 'Connection2'
, @v_return_code OUTPUT
, @v_return_msg OUTPUT
, @v_result_num OUTPUT
```

Teradata

Teradata Parameters: Ws_Load_Change

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN

© Copyright, WhereScape Software Ltd All rights reserved.

Parameter Name	Datatype	Mode
p_action	VARCHAR(64)	IN
p_table	VARCHAR(64)	IN
p_new_value	VARCHAR(255)	IN
p_return_code	VARCHAR(1)	OUT
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT

Teradata Examples: Ws_Load_Change

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_sequence integer;
DECLARE p_job_name varchar(256);
DECLARE p_task_name varchar(256);
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE p_return_msg varchar(256);
DECLARE p_status integer;
DECLARE v_result_num integer;
DECLARE v_return_code varchar(1);
DECLARE v_return_msg varchar(256);
CALL [METABASE].Ws_Load_Change
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, 'CONNECTION', 'load_customer', 'Connection2'
, v_return_code
, v_return_msg
, v_result_num
);
```

Oracle

Oracle Parameters: Ws_Load_Change

Callable Routine Type: FUNCTION.

Parameter Name	Datatype	Mode
p_sequence	NUMBER	IN
p_job_name	VARCHAR2	IN
p_task_name	VARCHAR2	IN
p_job_id	NUMBER	IN
p_task_id	NUMBER	IN
p_action	VARCHAR2	IN
p_table	VARCHAR2	IN
p_new_value	VARCHAR2	IN
p_return_code	VARCHAR2	OUT
p_return_msg	VARCHAR2	OUT
FUNCTION Return Value	NUMBER	OUT-Function

© Copyright, WhereScape Software Ltd All rights reserved.

Oracle Examples: Ws_Load_Change

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
p_sequence number;
p_job_name varchar2(256);
p_task_name varchar2(256);
p_job_id number;
p_task_id number;
p_return_msg varchar2(256);
p_status number;
v_result_num number;
v_return_code varchar2(1);
v_return_msg varchar2(256);
v_result_num := Ws_Load_Change
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, 'CONNECTION', 'load_customer', 'Connection2'
, v_return_code
, v_return_msg
);
```

DB2

DB2 Parameters: Ws_Load_Change

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_action	VARCHAR(64)	IN
p_table	VARCHAR(64)	IN
p_new_value	VARCHAR(255)	IN
p_return_code	VARCHAR(1)	OUT
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT

DB2 Examples: Ws_Load_Change

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_sequence integer;
DECLARE p_job_name varchar(256);
DECLARE p_task_name varchar(256);
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE p_return_msg varchar(256);
DECLARE p_status integer;
DECLARE v_result_num integer;
DECLARE v_return_code varchar(1);
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
DECLARE v_return_msg varchar(256);
CALL [METABASE].Ws_Load_Change
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, 'CONNECTION', 'load_customer', 'Connection2'
, v_return_code
, v_return_msg
, v_result_num
);
```

Ws_Maintain_Indexes

Synopsis

Drops and/or builds database indexes that are defined in the WhereScape RED metadata.

Description

Drops and/or builds indexes for a specified table or a specified index. Only indexes that are defined in the WhereScape RED metadata are supported. Typically, this routine is used by a WhereScape RED Scheduler and RED-generated procedures to automatically maintain indexes. However, it is also valid for user-defined custom procedures/scripts to execute this routine to control when indexes are dropped and/or created.

Input

Input	Description
Common Input	Includes all 5 inputs of the **Callable Routines Common Input**.
Table Name	Table Name to process the relevant indexes of. **Notes** The Table Name is ignored when the optional Index Name is specified.
Optional Build Arguments [ORACLE only]	Optional arguments related to building indexes **[ORACLE only]**: TABLESPACE=XYZ will override the default tablespace name and use the specified tablespace name. e.g. 'TABLESPACE=non_default_tablespace'. Typically the TABLESPACE argument is used to build indexes in a partition-specific tablespace.
Index Name	Optional Index Name to only process the specified index. When NOT specified all the relevant indexes of the table are processed. **Notes** • The Table Name is ignored when the Index Name is specified. • Must be specified to use the 'DROP' or 'BUILD' index actions.
Index Action	Action that specifies whether indexes are dropped or built and what types of indexes are applicable: • **DROP** - Drops the specified index (Index Name must be specified). • **DROP ALL** - Drops ALL the indexes of the table. • **PRE DROP** - Drops the indexes of the table that are defined as pre-drop. • **BUILD** - Builds the specified index (Index Name must be specified). Otherwise, build all the indexes of the table that were pre-dropped. • **BUILD ALL** - Builds ALL the indexes of the table.

Output

Output	Description
Result Number	Output Result Number: • 1 Success. • **-1 Warning.**

© Copyright, WhereScape Software Ltd All rights reserved.

Output	Description
	• -2 Error. • -3 Fatal/Unexpected Error.

Note

Ws_Maintain_Indexes does NOT include a Return Code or Return Message like most of the WhereScape RED Callable routines, but it does output a Result Number.

SQL Server

SQL Server Parameters: Ws_Maintain_Indexes

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
@p_sequence	INTEGER	IN
@p_job_name	VARCHAR(64)	IN
@p_task_name	VARCHAR(64)	IN
@p_job_id	INTEGER	IN
@p_task_id	INTEGER	IN
@p_table_name	VARCHAR(64)	IN
@p_parameter	VARCHAR(64)	IN
@p_index_name	VARCHAR(64)	IN
@p_option	VARCHAR(64)	IN
@p_result	INTEGER	OUT

SQL Server Examples: Ws_Maintain_Indexes

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE @p_sequence integer
DECLARE @p_job_name varchar(256)
DECLARE @p_task_name varchar(256)
DECLARE @p_job_id integer
DECLARE @p_task_id integer
DECLARE @p_return_msg varchar(256)
DECLARE @p_status integer
DECLARE @v_result_num integer
EXEC Ws_Maintain_Indexes
 @p_sequence, @p_job_name, @p_task_name, @p_job_id, @p_task_id
, 'load_customer', NULL, NULL, 'DROP ALL'
, @v_result_num OUTPUT
```

© Copyright, WhereScape Software Ltd All rights reserved.

Teradata

Teradata Parameters: Ws_Maintain_Indexes
Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_table_name	VARCHAR(64)	IN
p_parameter	VARCHAR(4000)	IN
p_index_name	VARCHAR(64)	IN
p_option	VARCHAR(20)	IN
p_result	INTEGER	OUT

Teradata Examples: Ws_Maintain_Indexes

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_sequence integer;
DECLARE p_job_name varchar(256);
DECLARE p_task_name varchar(256);
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE p_return_msg varchar(256);
DECLARE p_status integer;
DECLARE v_result_num integer;
CALL [METABASE].Ws_Maintain_Indexes
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, 'load_customer', NULL, NULL, 'DROP ALL'
, v_result_num
);
```

Oracle

Oracle Parameters: Ws_Maintain_Indexes
Callable Routine Type: FUNCTION.

Parameter Name	Datatype	Mode
p_sequence	NUMBER	IN
p_job_name	VARCHAR2	IN
p_task_name	VARCHAR2	IN
p_job_id	NUMBER	IN
p_task_id	NUMBER	IN
p_table_name	VARCHAR2	IN

© Copyright, WhereScape Software Ltd All rights reserved.

Parameter Name	Datatype	Mode
p_parameter	VARCHAR2	IN
p_index_name	VARCHAR2	IN
p_option	VARCHAR2	IN
FUNCTION Return Value	NUMBER	OUT-Function

Oracle Examples: Ws_Maintain_Indexes

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
p_sequence number;
p_job_name varchar2(256);
p_task_name varchar2(256);
p_job_id number;
p_task_id number;
p_return_msg varchar2(256);
p_status number;
v_result_num number;
v_result_num := Ws_Maintain_Indexes
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, 'load_customer', NULL, NULL, 'DROP ALL'
);
```

DB2

DB2 Parameters: Ws_Maintain_Indexes

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_table_name	VARCHAR(64)	IN
p_parameter	VARCHAR(4000)	IN
p_index_name	VARCHAR(64)	IN
p_option	VARCHAR(20)	IN
p_result	INTEGER	OUT

DB2 Examples: Ws_Maintain_Indexes

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_sequence integer;
DECLARE p_job_name varchar(256);
DECLARE p_task_name varchar(256);
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE p_return_msg varchar(256);
DECLARE p_status integer;
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
DECLARE v_result_num integer;
CALL [METABASE].Ws_Maintain_Indexes
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, 'load_customer', NULL, NULL, 'DROP ALL'
, v_result_num
);
```

Ws_Version_Clear

Synopsis

Purges metadata versions for all objects that do not meet the specified retention criteria.

Description

Deletes metadata versions for all objects that do not meet the specified retention criteria, which can be specified as the minimum number of versions to retain per object and/or the maximum age (in days) of versions to retain. For example, it is possible to specify that a minimum of 5 versions are retained for each object and/or that versions are retained for a maximum of 90 days.

Input

Input	Description
Common Input	Includes all 5 inputs of the **Callable Routines Common Input**.
Maximum Days to Retain	The maximum age (in days) of the versions to retain. If 90 days are retained, then all the versions that are older than 90 days are **purged/deleted** to reduce the number of versions to the specified maximum number of versions per object. If not specified, then the retention date of each version determines whether it is deleted.
Minimum Versions per Object to Retain	The minimum number of versions to retain for each object. If 5 is specified, then the last 5 versions are retained per object **regardless** of the specified maximum age to retain.
Options	Currently NOT used.

Output

Output	Description
Return Code	Output Return Code: • S Success. • E Error. • F Fatal/Unexpected Error.
Return Message	Output message indicating the action applied or the reason for no action.
Result Number	Output Result Number: • 1 Success. • -2 Error. • -3 Fatal/Unexpected Error.

SQL Server

SQL Server Parameters: Ws_Version_Clear

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
@p_sequence	INTEGER	IN
@p_job_name	VARCHAR(64)	IN
@p_task_name	VARCHAR(64)	IN

© Copyright, WhereScape Software Ltd All rights reserved.

Parameter Name	Datatype	Mode
@p_job_id	INTEGER	IN
@p_task_id	INTEGER	IN
@p_day_count	INTEGER	IN
@p_keep_count	INTEGER	IN
@p_options	VARCHAR(256)	IN
@p_return_code	VARCHAR(1)	OUT
@p_return_msg	VARCHAR(256)	OUT
@p_result	INTEGER	OUT

SQL Server Examples: Ws_Version_Clear

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE @p_sequence integer
DECLARE @p_job_name varchar(256)
DECLARE @p_task_name varchar(256)
DECLARE @p_job_id integer
DECLARE @p_task_id integer
DECLARE @p_return_msg varchar(256)
DECLARE @p_status integer
DECLARE @v_result_num integer
DECLARE @v_return_code varchar(1)
DECLARE @v_return_msg varchar(256)
EXEC Ws_Version_Clear
 @p_sequence, @p_job_name, @p_task_name, @p_job_id, @p_task_id
, 90, 5, NULL
, @v_return_code OUTPUT
, @v_return_msg OUTPUT
, @v_result_num OUTPUT
```

Teradata

Teradata Parameters: Ws_Version_Clear
Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_day_count	INTEGER	IN
p_keep_count	INTEGER	IN
p_options	VARCHAR(256)	IN
p_return_code	VARCHAR(1)	OUT
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT

Teradata Examples: Ws_Version_Clear

© Copyright, WhereScape Software Ltd All rights reserved.

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_sequence integer;
DECLARE p_job_name varchar(256);
DECLARE p_task_name varchar(256);
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE p_return_msg varchar(256);
DECLARE p_status integer;
DECLARE v_result_num integer;
DECLARE v_return_code varchar(1);
DECLARE v_return_msg varchar(256);
CALL [METABASE].Ws_Version_Clear
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, 90, 5, NULL
, v_return_code
, v_return_msg
, v_result_num
);
```

Oracle

Oracle Parameters: Ws_Version_Clear

Callable Routine Type: FUNCTION.

Parameter Name	Datatype	Mode
p_sequence	NUMBER	IN
p_job_name	VARCHAR2	IN
p_task_name	VARCHAR2	IN
p_job_id	NUMBER	IN
p_task_id	NUMBER	IN
p_day_count	NUMBER	IN
p_keep_count	NUMBER	IN
p_options	VARCHAR2	IN
p_return_code	VARCHAR2	OUT
p_return_msg	VARCHAR2	OUT
FUNCTION Return Value	NUMBER	OUT-Function

Oracle Examples: Ws_Version_Clear

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
p_sequence number;
p_job_name varchar2(256);
p_task_name varchar2(256);
p_job_id number;
p_task_id number;
p_return_msg varchar2(256);
p_status number;
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
v_result_num number;
v_return_code varchar2(1);
v_return_msg varchar2(256);
v_result_num := Ws_Version_Clear
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, 90, 5, NULL
, v_return_code
, v_return_msg
);
```

DB2

DB2 Parameters: Ws_Version_Clear

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_day_count	INTEGER	IN
p_keep_count	INTEGER	IN
p_options	VARCHAR(256)	IN
p_return_code	VARCHAR(1)	OUT
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT

DB2 Examples: Ws_Version_Clear

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_sequence integer;
DECLARE p_job_name varchar(256);
DECLARE p_task_name varchar(256);
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE p_return_msg varchar(256);
DECLARE p_status integer;
DECLARE v_result_num integer;
DECLARE v_return_code varchar(1);
DECLARE v_return_msg varchar(256);
CALL [METABASE].Ws_Version_Clear
( p_sequence, p_job_name, p_task_name, p_job_id, p_task_id
, 90, 5, NULL
, v_return_code
, v_return_msg
, v_result_num
);
```

© Copyright, WhereScape Software Ltd All rights reserved.

WsParameterRead

Synopsis

Returns the value and comment (for most RDBMS) of a WhereScape RED metadata Parameter.

Description

Returns the value and comment (for most RDBMS) of the specified parameter from the DSS_PARAMETER metadata table. For SQL Server, Teradata, and DB2 this routine is a PROCEDURE that returns both the parameter value and comment. However, for Oracle this routine is a FUNCTION that only returns the parameter value. For SQL Server, there is also a WsParameterReadF FUNCTION.

Typically, this routine is used by procedures to read information that is written by another process (automatically or manually via the RED **Tools > Parameters** menu item), which is external to the procedure.

Input

Input	Description
Parameter Name	The case-sensitive name of the WhereScape RED metadata parameter to be retrieved. The name must exactly match an existing parameter, otherwise a NULL value is returned.

Output

Output	Description
Parameter Value	The retrieved value of the parameter. Corresponds to the "Value" property that is visible and maintainable via **Tools > Parameters**.
Parameter Comments	The maximum number of versions to retain for each object. If 5 is specified then the last 5 versions are retained per object **regardless** of the specified maximum age to retain.

SQL Server

SQL Server Parameters: WsParameterRead

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
@p_parameter	VARCHAR(64)	IN
@p_value	VARCHAR(2000)	OUT
@p_comment	VARCHAR(256)	OUT

Note

There is also a corresponding WsParameterReadF FUNCTION available for SQL Server.

SQL Server Examples: WsParameterRead

```
DECLARE @v_current_date varchar(4000) -- Same length as DSS_PARAMETER.dss_parameter_value.
DECLARE @v_comment varchar(256)
EXEC WsParameterRead 'CURRENT_DATE', @v_current_date OUTPUT, @v_comment OUTPUT;
```

Teradata

Teradata Parameters: WsParameterRead

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_parameter	VARCHAR(64)	IN
p_value	VARCHAR(**2000**)	OUT

© Copyright, WhereScape Software Ltd All rights reserved.

Parameter Name	Datatype	Mode
p_comment	VARCHAR(256)	OUT

Teradata Examples: WsParameterRead

```
DECLARE v_current_date varchar(4000); -- Same length as DSS_PARAMETER.dss_parameter_value.
DECLARE v_comment varchar(256);
CALL [METABASE].WsParameterRead('CURRENT_DATE',v_current_date,v_comment);
```

Oracle

Oracle Parameters: WsParameterRead

Callable Routine Type: FUNCTION.

Parameter Name	Datatype	Mode
v_parameter	VARCHAR2	IN
FUNCTION Return Value	VARCHAR2	OUT-Function

Oracle Examples: WsParameterRead

```
v_current_date varchar2(4000); -- Same length as DSS_PARAMETER.dss_parameter_value.
v_current_date := WsParameterRead('CURRENT_DATE'); -- NOTE: Does NOT support comment
output.
```

DB2

DB2 Parameters: WsParameterReadG

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_parameter	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_value	VARCHAR(2000)	OUT

DB2 Examples: WsParameterReadG

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE v_source_name varchar(256);
CALL [METABASE].WsParameterReadG('$$SOURCE_TABLE', p_job_id, p_task_id, v_source_name);
```

© Copyright, WhereScape Software Ltd All rights reserved.

WsParameterReadF

Synopsis

Returns the value of a WhereScape RED metadata Parameter **[SQL Server only]**.

Description

Returns the value of the specified parameter from the DSS_PARAMETER metadata table. This routine is a FUNCTION that is only available for SQL Server. For SQL Server, Teradata, and DB2 there is a WsParameterRead PROCEDURE that returns both the parameter value and comment. For Oracle the WsParameterRead FUNCTION is equivalent to the SQL Server WsParameterReadF FUNCTION.

Typically, this routine is used by procedures to read information that is written by another process (automatically or manually via the RED **Tools > Parameters** menu item), which is external to the procedure.

Input

Input	Description
Parameter Name	The name of the WhereScape RED metadata parameter to be retrieved. The case-sensitive name must exactly match an existing parameter, otherwise a NULL value is returned.

Output

Output	Description
Parameter Value	The value of the parameter. Corresponds to the **Value** property that is visible and maintainable via **Tools > Parameters**.

SQL Server

SQL Server Parameters: WsParameterReadF

Callable Routine Type: FUNCTION.

Parameter Name	Datatype	Mode
@p_parameter	VARCHAR(64)	IN
FUNCTION Return Value	VARCHAR(4000)	OUT-Function

Note

There is also a corresponding WsParameterRead PROCEDURE available for SQL Server.

SQL Server Examples: WsParameterReadF

```
DECLARE @v_current_date varchar(4000)
SELECT @v_current_date = dbo.WsParameterReadF('CURRENT_DATE')
```

© Copyright, WhereScape Software Ltd All rights reserved.

WsParameterReadG

Synopsis

Returns the value of a "global" WhereScape RED metadata Parameter that relates to a load table.

Description

Returns the value of an internal parameter that is defined and populated by WhereScape RED, which is available to a procedure that is currently processing a load table.

The supported parameters are $$TABLE_NAME and $$SOURCE_TABLE.

Input

Input	Description
Global Parameter Name	The supported global parameters are: • **$$TABLE_NAME** returns the Load Table Name that the procedure is executing against. Only available for load tables. • **$$SOURCE_TABLE** returns the maximum value of the Source Table property from the columns of the Load Table that the procedure is executing against. Only available for load tables. **Note** Typically, a Load Table has a single Source Table but if it has multiple sources then the maximum (alphabetically) Source Table Name will be returned.
Job Identifier	Unique identifier of the **held or scheduled job** that the running job is a specific instance of. When invoked from a WhereScape RED Scheduler the routine will be passed the parameter argument. When invoked manually or externally to the WhereScape RED Scheduler, it is recommended to use 0 (zero).
Task or Object Identifier	Unique identifier of the **running task** (of a running job) that executed the routine. When invoked from a WhereScape RED Scheduler the routine will be passed the parameter argument. When invoked manually or externally to the WhereScape RED Scheduler, it should be the **object key**.

Output

Output	Description
Result Table Name	The requested Load Table Name or Source Table Name.

SQL Server

SQL Server Parameters: WsParameterReadG

Callable Routine Type: FUNCTION.

Parameter Name	Datatype	Mode
@p_parameter	VARCHAR(64)	IN
@p_job_id	INTEGER	IN
@p_task_id	INTEGER	IN
FUNCTION Return Value	VARCHAR(4000)	OUT-Function

SQL Server Examples: WsParameterReadG

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE @p_job_id integer
DECLARE @p_task_id integer
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
DECLARE @v_source_name varchar(64)
SELECT @v_source_name = dbo.WsParameterReadG('$$SOURCE_TABLE', @p_job_id, @p_task_id) --
Needs dbo.
```

Teradata

Teradata Parameters: WsParameterReadG

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_parameter	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_value	VARCHAR(2000)	OUT

Teradata Examples: WsParameterReadG

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE v_source_name varchar(256);
CALL [METABASE].WsParameterReadG('$$SOURCE_TABLE', p_job_id, p_task_id, v_source_name);
```

Oracle

Oracle Parameters: WsParameterReadG

Callable Routine Type: FUNCTION.

Parameter Name	Datatype	Mode
p_parameter	VARCHAR2	IN
p_job_id	NUMBER	IN
p_task_id	NUMBER	IN
FUNCTION Return Value	VARCHAR2	OUT-Function

Oracle Examples: WsParameterReadG

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
p_job_id number;
p_task_id number;
v_source_name varchar2(64);
v_source_name := WsParameterReadG('$$SOURCE_TABLE', p_job_id, p_task_id);
```

DB2

DB2 Parameters: WsParameterReadG

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_parameter	VARCHAR(64)	IN
p_job_id	INTEGER	IN

© Copyright, WhereScape Software Ltd All rights reserved.

Parameter Name	Datatype	Mode
p_task_id	INTEGER	IN
p_value	VARCHAR(2000)	OUT

DB2 Examples: WsParameterReadG

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE v_source_name varchar(256);
CALL [METABASE].WsParameterReadG('$$SOURCE_TABLE', p_job_id, p_task_id, v_source_name);
```

WsParameterWrite

Synopsis

Updates the value and comment of a WhereScape RED metadata Parameter or creates it.

Description

Updates the value and comment of the specified parameter in the DSS_PARAMETER metadata table. If the specified parameter is not found, then it is added.

Typically, this routine is used by procedures to write information that is read by another process (automatically or manually via the RED **Tools > Parameters** menu item), which is external to the procedure.

Input

Input	Description
Parameter Name	The case-sensitive name of the WhereScape RED metadata parameter to be updated or added.
Parameter Value	The new value of the parameter to be assigned. Corresponds to the "Value" property that is visible and maintainable via **Tools > Parameters**.
Parameter Comments	The new comments of the parameter to be assigned. Corresponds to the "Comments" property that is visible and maintainable via **Tools > Parameters**. The parameter comments will not be modified if a NULL value is specified.

Output

Output	Description
Result Number [SQL SERVER & ORACLE only]	Output Result Number **[SQL SERVER & ORACLE only]**: • **1Metadata Parameter Updated.** • **2Metadata Parameter Inserted.** • -3Fatal/Unexpected Error.

SQL Server

SQL Server Parameters: WsParameterWrite

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
@p_parameter	VARCHAR(64)	IN
@p_value	VARCHAR(2000)	IN
@p_comment	VARCHAR(256)	IN

SQL Server Examples: WsParameterWrite

```
-- Example 1: Result Number (procedure return code) NOT captured.
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
EXEC WsParameterWrite 'LAST_INVOICE_ID', '123456', 'The last invoice id loaded'

-- Example 2: Result Number (procedure return code) captured.
DECLARE @v_result_num           integer
EXEC @v_result_num = WsParameterWrite 'LAST_INVOICE_ID', '123456', 'The last invoice id
loaded'
```

Teradata

Teradata Parameters: WsParameterWrite

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_parameter	VARCHAR(64)	IN
p_value	VARCHAR(2000)	IN
p_comment	VARCHAR(256)	IN

Teradata Examples: WsParameterWrite

```
CALL [METABASE].WsParameterWrite('LAST_INVOICE_ID', '123456', 'The last invoice id
loaded');
```

Oracle

Oracle Parameters: WsParameterWrite

Callable Routine Type: FUNCTION.

Parameter Name	Datatype	Mode
v_parameter	VARCHAR2	IN
v_value	VARCHAR2	IN
v_comment	VARCHAR2	IN
FUNCTION Return Value	VARCHAR2	OUT-Function

Oracle Examples: WsParameterWrite

```
v_result_num number;
v_result_num := WsParameterWrite('LAST_INVOICE_ID', '123456', 'The last invoice id
loaded');
```

DB2

DB2 Parameters: WsParameterWrite

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_parameter	VARCHAR(64)	IN
p_value	VARCHAR(2000)	IN
p_comment	VARCHAR(256)	IN

DB2 Examples: WsParameterWrite

```
CALL [METABASE].WsParameterWrite('LAST_INVOICE_ID', '123456', 'The last invoice id
loaded');
```

© Copyright, WhereScape Software Ltd All rights reserved.

WsWrkAudit

Synopsis

Records a message in the Audit Log.

Description

Adds the specified message to the WS_WRK_AUDIT_LOG workflow metadata table, which is referred to as the Audit Log or Audit Trail. A variety of message types are supported such as Information, Warning, and Error that are included in the corresponding message type counts for the task and job. Audit Log messages are accessible via the "Scheduler" tab/window and/or the WS_ADMIN_V_AUDIT view of the WS_WRK_AUDIT_LOG table.

Note

Both the Audit Log and Error/Detail Log support similar information and in user-defined custom procedures, either or both logs can be used. However in RED-generated procedures/scripts, the Audit Log is used for higher-level or summary messages while the Error/Detail Log is used for more detailed supporting information.

Input

Input	Description
Common Input	Includes all 5 inputs of the **Callable Routines Common Input**. **Note** Refer to the RDBMS-specific parameters for the relative positions (they are NOT declared as the first parameters).
Audit Message Type Code	Audit Message Type Code: • B - Beginning of a Job or Task. • I - Information. • S - Success. • W - Warning. • E - Error. • F - Fatal Error.
Audit Message Text	Custom message text to be recorded in the WhereScape RED Audit Log.
RDBMS Code	RDBMS-specific message code. e.g. The Oracle special variable SQLCODE. It is optional but recommended to populate this when an error occurs.
RDBMS Message	RDBMS-specific message. e.g. The Oracle special variable SQLERRM. It is optional but recommended to populate this when an error occurs.

Output

Output	Description
Result Number	Output Result Number: • 1 Success. • -3 Error.

SQL Server

SQL Server Parameters: WsWrkAudit

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
@p_status_code	VARCHAR(1)	IN
@p_job_name	VARCHAR(64)	IN

© Copyright, WhereScape Software Ltd All rights reserved.

Parameter Name	Datatype	Mode
@p_task_name	VARCHAR(64)	IN
@p_sequence	INTEGER	IN
@p_message	VARCHAR(256)	IN
@p_db_code	VARCHAR(10)	IN
@p_db_msg	VARCHAR(256)	IN
@p_task_key	INTEGER	IN
@p_job_key	INTEGER	IN

SQL Server Examples: WsWrkAudit

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE @p_sequence integer
DECLARE @p_job_name varchar(256)
DECLARE @p_task_name varchar(256)
DECLARE @p_job_id integer
DECLARE @p_task_id integer
DECLARE @p_return_msg varchar(256)
DECLARE @p_status integer
DECLARE @v_result_num integer
EXEC @v_result_num = WsWrkAudit
 'I', @p_job_name, @p_task_name, @p_sequence
, 'The task has started.'
, NULL
, NULL
, @p_task_id
, @p_job_id
```

Teradata

Teradata Parameters: WsWrkAudit

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_status_code	VARCHAR(1)	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_sequence	INTEGER	IN
p_message	VARCHAR(255)	IN
p_db_code	VARCHAR(10)	IN
p_db_msg	VARCHAR(255)	IN
p_task_key	INTEGER	IN
p_job_key	INTEGER	IN

Teradata Examples: WsWrkAudit

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_sequence integer;
DECLARE p_job_name varchar(256);
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
DECLARE p_task_name varchar(256);
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE p_return_msg varchar(256);
DECLARE p_status integer;
CALL [METABASE].WsWrkAudit
( 'I', p_job_name, p_task_name, p_sequence
, 'The task has started.'
, NULL
, NULL
, p_task_id
, p_job_id
);
```

Oracle

Oracle Parameters: WsWrkAudit

Callable Routine Type: FUNCTION.

Parameter Name	Datatype	Mode
v_status_code	VARCHAR2	IN
v_job_name	VARCHAR2	IN
v_task_name	VARCHAR2	IN
v_sequence	NUMBER	IN
v_message	VARCHAR2	IN
v_db_code	VARCHAR2	IN
v_db_msg	VARCHAR2	IN
v_task_key	NUMBER	IN
v_job_key	NUMBER	IN
FUNCTION Return Value	NUMBER	OUT-Function

Oracle Examples: WsWrkAudit

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
p_sequence number;
p_job_name varchar2(256);
p_task_name varchar2(256);
p_job_id number;
p_task_id number;
p_return_msg varchar2(256);
p_status number;
v_result_num number;
v_result_num := WsWrkAudit
( 'I', p_job_name, p_task_name, p_sequence
, 'The task has started.'
, NULL
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
, NULL
, p_task_id
, p_job_id
);
```

DB2

DB2 Parameters: WsWrkAudit

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
v_status_code	CHARACTER	IN
v_job_name	VARCHAR(64)	IN
v_task_name	VARCHAR(64)	IN
v_sequence	INTEGER	IN
v_message	VARCHAR(256)	IN
v_db_code	VARCHAR(10)	IN
v_db_msg	VARCHAR(256)	IN
v_task_key	INTEGER	IN
v_job_key	INTEGER	IN

DB2 Examples: WsWrkAudit

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_sequence integer;
DECLARE p_job_name varchar(256);
DECLARE p_task_name varchar(256);
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE p_return_msg varchar(256);
DECLARE p_status integer;
CALL [METABASE].WsWrkAudit
( 'I', p_job_name, p_task_name, p_sequence
, 'The task has started.'
, NULL
, NULL
, p_task_id
, p_job_id
);
```

© Copyright, WhereScape Software Ltd All rights reserved.

WsWrkAuditBulk

Synopsis

Records multiple messages in the Audit Log.

Description

Adds the specified multiple messages to the WS_WRK_AUDIT_LOG workflow metadata table, which is referred to as the Audit Log or Audit Trail. A variety of message types are supported, such as Information, Warning, and Error that are included in the corresponding message type counts for the task and job. Audit Log messages are accessible via the "Scheduler" tab/window and/or the WS_ADMIN_V_AUDIT view of the WS_WRK_AUDIT_LOG table.

Note

Both the Audit Log and Error/Detail Log support similar information and in user-defined custom procedures, either or both logs can be used. However in RED-generated procedures/scripts, the Audit Log is used for higher-level or summary messages while the Error/Detail Log is used for more detailed supporting information.

Input

Input	Description
Common Input	Includes all 5 inputs of the **Callable Routines Common Input**. **Note** Refer to the RDBMS-specific parameters for the relative positions (they are NOT declared as the first parameters).
Audit Message Type Code	Audit Message Type Code: • B - Beginning of a Job or Task. • I - Information. • S - Success. • W - Warning. • E - Error. • F - Fatal Error.
Audit Message(s) Text	Custom message(s) text to be recorded in the WhereScape RED Audit Log. Multiple messages can be specified but each is limited to 256 characters. Each message must be separated by either a new-line (ASCII 10) or tilde (~) character. e.g. Message1~Message2~Message3 will create 3 messages.
RDBMS Code	RDBMS-specific message code. e.g. The Oracle special variable SQLCODE. It is optional but recommended to populate this when an error occurs.
RDBMS Message	RDBMS-specific message. e.g. The Oracle special variable SQLERRM. It is optional but recommended to populate this when an error occurs.

Output

Output	Description
Result Number	**Note** Not provided for all RDBMS. Output Result Number: • 1 Success. • -2 Error • -3 Fatal/Unexpected Error.

© Copyright, WhereScape Software Ltd All rights reserved.

SQL Server

SQL Server Parameters: WsWrkAuditBulk

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
@p_status_code	VARCHAR(1)	IN
@p_job_name	VARCHAR(64)	IN
@p_task_name	VARCHAR(64)	IN
@p_sequence	INTEGER	IN
@p_message	VARCHAR(4000)	IN
@p_db_code	VARCHAR(10)	IN
@p_db_msg	VARCHAR(256)	IN
@p_task_key	INTEGER	IN
@p_job_key	INTEGER	IN

SQL Server Examples: WsWrkAuditBulk

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE @p_sequence integer
DECLARE @p_job_name varchar(256)
DECLARE @p_task_name varchar(256)
DECLARE @p_job_id integer
DECLARE @p_task_id integer
DECLARE @p_return_msg varchar(256)
DECLARE @p_status integer
DECLARE @v_result_num integer
EXEC @v_result_num = WsWrkAuditBulk
 'I', @p_job_name, @p_task_name, @p_sequence
, 'Message1~Message2~Message3'
, NULL
, NULL
, @p_job_id
, @p_task_id
```

Teradata

Teradata Parameters: WsWrkAuditBulk

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_status_code	VARCHAR(64)	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_sequence	INTEGER	IN
p_message	VARCHAR(61440)	IN
p_db_code	VARCHAR(10)	IN
p_db_msg	VARCHAR(256)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN

© Copyright, WhereScape Software Ltd All rights reserved.

Parameter Name	Datatype	Mode
p_result	INTEGER	OUT

Teradata Examples: WsWrkAuditBulk

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_sequence integer;
DECLARE p_job_name varchar(256);
DECLARE p_task_name varchar(256);
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE p_return_msg varchar(256);
DECLARE p_status integer;
DECLARE v_result_num integer;
CALL [METABASE].WsWrkAuditBulk
( 'I', p_job_name, p_task_name, p_sequence
, 'Message1~Message2~Message3'
, NULL
, NULL
, p_job_id --### NOTE order.
, p_task_id --### NOTE order.
, v_result_num
);
```

Oracle

Oracle Parameters: WsWrkAuditBulk

Callable Routine Type: FUNCTION.

Parameter Name	Datatype	Mode
v_status_code	VARCHAR2	IN
v_job_name	VARCHAR2	IN
v_task_name	VARCHAR2	IN
v_sequence	NUMBER	IN
v_message	VARCHAR2	IN
v_db_code	VARCHAR2	IN
v_db_msg	VARCHAR2	IN
v_task_key	NUMBER	IN
v_job_key	NUMBER	IN
FUNCTION Return Value	NUMBER	OUT-Function

Oracle Examples: WsWrkAuditBulk

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
p_sequence number;
p_job_name varchar2(256);
p_task_name varchar2(256);
p_job_id number;
p_task_id number;
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
p_return_msg varchar2(256);
p_status number;
v_result_num number;
v_result_num := WsWrkAuditBulk
( 'I', p_job_name, p_task_name, p_sequence
, 'Message1~Message2~Message3'
, NULL
, NULL
, p_task_id
, p_job_id
);
```

DB2

DB2 Parameters: WsWrkAuditBulk

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_status_code	VARCHAR(64)	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_sequence	INTEGER	IN
p_message	VARCHAR(32672)	IN
p_db_code	VARCHAR(10)	IN
p_db_msg	VARCHAR(256)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_result	INTEGER	OUT

DB2 Examples: WsWrkAuditBulk

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_sequence integer;
DECLARE p_job_name varchar(256);
DECLARE p_task_name varchar(256);
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE p_return_msg varchar(256);
DECLARE p_status integer;
DECLARE v_result_num integer;
CALL [METABASE].WsWrkAuditBulk
( 'I', p_job_name, p_task_name, p_sequence
, 'Message1~Message2~Message3'
, NULL
, NULL
, p_job_id --### NOTE order.
, p_task_id --### NOTE order.
, v_result_num
);
```

© Copyright, WhereScape Software Ltd All rights reserved.

WsWrkError

Synopsis

Records a message in the Error/Detail Log.

Description

Adds the specified message to the WS_WRK_ERROR_LOG workflow metadata table, which is referred to as the Error Log or Detail Log. A variety of message types are supported such as Information, Warning, and Error that are included in the "detail" message counts for the task and job (viewable via the "Scheduler" tab/window). Error/Detail Log messages are accessible via the "Scheduler" tab/window and/or the WS_ADMIN_V_ERROR view of the WS_WRK_ERROR_LOG table.

Note

Both the Audit Log and Error/Detail Log support similar information and in user-defined custom procedures, either or both logs can be used. However in RED-generated procedures/scripts, the Audit Log is used for higher-level or summary messages while the Error/Detail Log is used for more detailed supporting information.

Input

Input	Description
Common Input	Includes all 5 inputs of the **Callable Routines Common Input**. **Note** Refer to the RDBMS-specific parameters for the relative positions (they are NOT declared as the first parameters).
Error/Detail Message Type Code	Error/Detail Message Type Code: • E - Error. • I - Information. • W - Warning.
Error/Detail Message Text	Custom message text to be recorded in the WhereScape RED Error/Detail Log.
RDBMS Code	RDBMS-specific message code. e.g. The Oracle special variable SQLCODE. It is optional but recommended to populate this when an error occurs.
RDBMS Message	RDBMS-specific message. e.g. The Oracle special variable SQLERRM. It is optional but recommended to populate this when an error occurs.
Custom Message Type Code	Custom Message Type Code. For custom usage and has no meaning within the WhereScape RED metadata.

Output

Output	Description
Result Number	Output Result Number: • 1 Success. • -3 Error.

© Copyright, WhereScape Software Ltd All rights reserved.

SQL Server

SQL Server Parameters: WsWrkError

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
@p_status_code	VARCHAR(1)	IN
@p_job_name	VARCHAR(64)	IN
@p_task_name	VARCHAR(64)	IN
@p_sequence	INTEGER	IN
@p_message	VARCHAR(256)	IN
@p_db_code	VARCHAR(10)	IN
@p_db_msg	VARCHAR(256)	IN
@p_task_key	INTEGER	IN
@p_job_key	INTEGER	IN
@p_msg_type	VARCHAR(10)	IN

SQL Server Examples: WsWrkError

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE @p_sequence integer
DECLARE @p_job_name varchar(256)
DECLARE @p_task_name varchar(256)
DECLARE @p_job_id integer
DECLARE @p_task_id integer
DECLARE @p_return_msg varchar(256)
DECLARE @p_status integer
DECLARE @v_result_num integer
EXEC @v_result_num = WsWrkError
 'I', @p_job_name, @p_task_name, @p_sequence
, 'This is an INFO message in the Error/Detail Log.'
, NULL
, NULL
, @p_task_id
, @p_job_id
, NULL
```

Teradata

Teradata Parameters: WsWrkError

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_status_code	VARCHAR(1)	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_sequence	INTEGER	IN
p_message	VARCHAR(255)	IN
p_db_code	VARCHAR(10)	IN

© Copyright, WhereScape Software Ltd All rights reserved.

Parameter Name	Datatype	Mode
p_db_msg	VARCHAR(255)	IN
p_task_key	INTEGER	IN
p_job_key	INTEGER	IN
p_msg_type	VARCHAR(10)	IN

Teradata Examples: WsWrkError

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_sequence integer;
DECLARE p_job_name varchar(256);
DECLARE p_task_name varchar(256);
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE p_return_msg varchar(256);
DECLARE p_status integer;
CALL [METABASE].WsWrkError
( 'I', p_job_name, p_task_name, p_sequence
, 'This is an INFO message in the Error/Detail Log.'
, NULL
, NULL
, p_task_id
, p_job_id
, NULL
);
```

Oracle

Oracle Parameters: WsWrkError

Callable Routine Type: FUNCTION.

Parameter Name	Datatype	Mode
v_status_code	VARCHAR2	IN
v_job_name	VARCHAR2	IN
v_task_name	VARCHAR2	IN
v_sequence	NUMBER	IN
v_message	VARCHAR2	IN
v_db_code	VARCHAR2	IN
v_db_msg	VARCHAR2	IN
v_task_key	NUMBER	IN
v_job_key	NUMBER	IN
v_msg_type	VARCHAR2	IN
FUNCTION Return Value	NUMBER	OUT-Function

Oracle Examples: WsWrkError

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
p_sequence number;
p_job_name varchar2(256);
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
p_task_name varchar2(256);
p_job_id number;
p_task_id number;
p_return_msg varchar2(256);
p_status number;
v_result_num number;
v_result_num := WsWrkError
( 'I', p_job_name, p_task_name, p_sequence
, 'This is an INFO message in the Error/Detail Log.'
, NULL
, NULL
, p_task_id
, p_job_id
, NULL
);
```

DB2

DB2 Parameters: WsWrkError

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
v_status_code	CHARACTER	IN
v_job_name	VARCHAR(64)	IN
v_task_name	VARCHAR(64)	IN
v_sequence	INTEGER	IN
v_message	VARCHAR(256)	IN
v_db_code	VARCHAR(10)	IN
v_db_msg	VARCHAR(256)	IN
v_task_key	INTEGER	IN
v_job_key	INTEGER	IN
v_msg_type	VARCHAR(10)	IN

DB2 Examples: WsWrkError

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_sequence integer;
DECLARE p_job_name varchar(256);
DECLARE p_task_name varchar(256);
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE p_return_msg varchar(256);
DECLARE p_status integer;
CALL [METABASE].WsWrkError
( 'I', p_job_name, p_task_name, p_sequence
, 'This is an INFO message in the Error/Detail Log.'
, NULL
, NULL
, p_task_id
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
, p_job_id
, NULL
);
```

WsWrkErrorBulk

Synopsis

Records multiple messages in the Error/Detail Log.

Description

Adds the specified multiple messages to the WS_WRK_ERROR_LOG workflow metadata table, which is referred to as the Error Log or Detail Log. A variety of message types are supported such as Information, Warning, and Error that are included in the "detail" message counts for the task and job (viewable via the "Scheduler" tab/window). Error/Detail Log messages are accessible via the "Scheduler" tab/window and/or the WS_ADMIN_V_ERROR view of the WS_WRK_ERROR_LOG table.

Note

Both the Audit Log and Error/Detail Log support similar information and in user-defined custom procedures either or both logs can be used. However in RED-generated procedures/scripts, the Audit Log is used for higher-level or summary messages while the Error/Detail Log is used for more detailed supporting information.

Input

Input	Description
Common Input	Includes all 5 inputs of the **Callable Routines Common Input**. **Note** Refer to the RDBMS-specific parameters for the relative positions (they are NOT declared as the first parameters).
Error/Detail Message Type Code	Error/Detail Message Type Code: • E - Error. • I - Information. • W - Warning.
Error/Detail Message(s) Text	Custom message(s) text to be recorded in the WhereScape RED Error/Detail Log. Multiple messages can be specified but each is limited to 256 characters. Each message must be separated by either a new-line (ASCII 10) or tilde (~) character. e.g. Message1~Message2~Message3 will create 3 messages.
RDBMS Code	RDBMS-specific message code. e.g. The Oracle special variable SQLCODE. It is optional but recommended to populate this when an error occurs.
RDBMS Message	RDBMS-specific message. e.g. The Oracle special variable SQLERRM. It is optional but recommended to populate this when an error occurs.
Custom Message Type Code	Custom Message Type Code. For custom usage and has no meaning within the WhereScape RED metadata.

Output

Output	Description
Result Number	**Note** NOT provided for all RDBMS. Output Result Number: • 1 Success. • -2 Error. • -3 Fatal/Unexpected Error.

© Copyright, WhereScape Software Ltd All rights reserved.

SQL Server

SQL Server Parameters: WsWrkErrorBulk

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
@p_status_code	VARCHAR(1)	IN
@p_job_name	VARCHAR(64)	IN
@p_task_name	VARCHAR(64)	IN
@p_sequence	INTEGER	IN
@p_message	VARCHAR(4000)	IN
@p_db_code	VARCHAR(10)	IN
@p_db_msg	VARCHAR(256)	IN
@p_task_key	INTEGER	IN
@p_job_key	INTEGER	IN
@p_msg_type	VARCHAR(10)	IN

SQL Server Examples: WsWrkErrorBulk

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE @p_sequence integer
DECLARE @p_job_name varchar(256)
DECLARE @p_task_name varchar(256)
DECLARE @p_job_id integer
DECLARE @p_task_id integer
DECLARE @p_return_msg varchar(256)
DECLARE @p_status integer
DECLARE @v_result_num integer
EXEC @v_result_num = WsWrkErrorBulk
 'I', @p_job_name, @p_task_name, @p_sequence
, 'Message1~Message2~Message3'
, NULL
, NULL
, @p_job_id
, @p_task_id
, NULL
```

Teradata

Teradata Parameters: WsWrkErrorBulk

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_status_code	VARCHAR(64)	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_sequence	INTEGER	IN
p_message	VARCHAR(61440)	IN
p_db_code	VARCHAR(10)	IN
p_db_msg	VARCHAR(256)	IN

© Copyright, WhereScape Software Ltd All rights reserved.

Parameter Name	Datatype	Mode
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_msg_type	VARCHAR(10)	IN
p_result	INTEGER	OUT

Teradata Examples: WsWrkErrorBulk

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_sequence integer;
DECLARE p_job_name varchar(256);
DECLARE p_task_name varchar(256);
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE p_return_msg varchar(256);
DECLARE p_status integer;
DECLARE v_result_num integer;
CALL [METABASE].WsWrkErrorBulk
( 'I', p_job_name, p_task_name, p_sequence
, 'Message1~Message2~Message3'
, NULL
, NULL
, p_job_id --### NOTE order.
, p_task_id --### NOTE order.
, NULL
, v_result_num
);
```

Oracle

Oracle Parameters: WsWrkErrorBulk

Callable Routine Type: FUNCTION.

Parameter Name	Datatype	Mode
v_status_code	VARCHAR2	IN
v_job_name	VARCHAR2	IN
v_task_name	VARCHAR2	IN
v_sequence	NUMBER	IN
v_message	VARCHAR2	IN
v_db_code	VARCHAR2	IN
v_db_msg	VARCHAR2	IN
v_task_key	NUMBER	IN
v_job_key	NUMBER	IN
v_msg_type	VARCHAR2	IN
FUNCTION Return Value	NUMBER	OUT-Function

© Copyright, WhereScape Software Ltd All rights reserved.

Oracle Examples: WsWrkErrorBulk

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
p_sequence number;
p_job_name varchar2(256);
p_task_name varchar2(256);
p_job_id number;
p_task_id number;
p_return_msg varchar2(256);
p_status number;
v_result_num number;
v_result_num := WsWrkErrorBulk
( 'I', p_job_name, p_task_name, p_sequence
, 'Message1~Message2~Message3'
, NULL
, NULL
, p_task_id
, p_job_id
, NULL
);
```

DB2

DB2 Parameters: WsWrkErrorBulk

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_status_code	VARCHAR(64)	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_sequence	INTEGER	IN
p_message	VARCHAR(32672)	IN
p_db_code	VARCHAR(10)	IN
p_db_msg	VARCHAR(256)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_msg_type	VARCHAR(10)	IN
p_result	INTEGER	OUT

DB2 Examples: WsWrkErrorBulk

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_sequence integer;
DECLARE p_job_name varchar(256);
DECLARE p_task_name varchar(256);
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE p_return_msg varchar(256);
DECLARE p_status integer;
DECLARE v_result_num integer;
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
CALL [METABASE].WsWrkErrorBulk
( 'I', p_job_name, p_task_name, p_sequence
, 'Message1~Message2~Message3'
, NULL
, NULL
, p_job_id --### NOTE order.
, p_task_id --### NOTE order.
, NULL
, v_result_num
);
```

WsWrkTask

Synopsis

Updates row counts for a task in the Task Log.

Description

Updates row counts for the specified task in the Task Log. Task Log messages (and row counts) are accessible via the "Scheduler" tab/window and/or the WS_ADMIN_V_TASK view of the WS_WRK_TASK_RUN and WS_WRK_TASK_LOG tables.

This routine is intended to be executed by a task of a job since it requires a valid job, task, and job sequence number that are provided by a WhereScape RED Scheduler.

Input

Input	Description
Common Input	Includes **3 inputs** of the **Callable Routines Common Input**.
Inserted Row Count	The number of rows inserted by the task.
Updated Row Count	The number of rows updated by the task.
Replaced Row Count	The number of rows replaced by the task.
Deleted Row Count	The number of rows deleted by the task.
Discarded Row Count	The number of rows discarded by the task.
Rejected Row Count	The number of rows rejected by the task.
Error Row Count	The number of rows with an error that were failed by the task.

Output

Output	Description
Result Number	Output Result Number: • **0 Success.** • **-1 Warning.** • -3 Fatal/Unexpected Error.

© Copyright, WhereScape Software Ltd All rights reserved.

SQL Server

SQL Server Parameters: WsWrkTask

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
@p_job_key	INTEGER	IN
@p_task_key	INTEGER	IN
@p_sequence	INTEGER	IN
@p_inserted	INTEGER	IN
@p_updated	INTEGER	IN
@p_replaced	INTEGER	IN
@p_deleted	INTEGER	IN
@p_discarded	INTEGER	IN
@p_rejected	INTEGER	IN
@p_errored	INTEGER	IN

SQL Server Examples: WsWrkTask

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE @p_sequence integer
DECLARE @p_job_name varchar(256)
DECLARE @p_task_name varchar(256)
DECLARE @p_job_id integer
DECLARE @p_task_id integer
DECLARE @p_return_msg varchar(256)
DECLARE @p_status integer
DECLARE @v_result_num integer
EXEC WsWrkTask
 @p_job_id, @p_task_id, @p_sequence
, 27, 30, 0, 0, 0, 0, 0
```

Teradata

Teradata Parameters: WsWrkTask

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_job_key	INTEGER	IN
p_task_key	INTEGER	IN
p_sequence	INTEGER	IN
p_inserted	INTEGER	IN
p_updated	INTEGER	IN
p_replaced	INTEGER	IN
p_deleted	INTEGER	IN
p_discarded	INTEGER	IN
p_rejected	INTEGER	IN
p_errored	INTEGER	IN

Teradata Examples: WsWrkTask

© Copyright, WhereScape Software Ltd All rights reserved.

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_sequence integer;
DECLARE p_job_name varchar(256);
DECLARE p_task_name varchar(256);
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE p_return_msg varchar(256);
DECLARE p_status integer;
DECLARE v_insert_count integer;
DECLARE v_update_count integer;
CALL [METABASE].WsWrkTask
( p_job_id, p_task_id, p_sequence
, v_insert_count, v_update_count, 0, 0, 0, 0, 0
);
```

Oracle

Oracle Parameters: WsWrkTask

Callable Routine Type: FUNCTION.

Parameter Name	Datatype	Mode
v_job_key	NUMBER	IN
v_task_key	NUMBER	IN
v_sequence	NUMBER	IN
v_inserted	NUMBER	IN
v_updated	NUMBER	IN
v_replaced	NUMBER	IN
v_deleted	NUMBER	IN
v_discarded	NUMBER	IN
v_rejected	NUMBER	IN
v_errored	NUMBER	IN
FUNCTION Return Value	NUMBER	OUT-Function

Oracle Examples: WsWrkTask

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
p_sequence number;
p_job_name varchar2(256);
p_task_name varchar2(256);
p_job_id number;
p_task_id number;
p_return_msg varchar2(256);
p_status number;
v_result_num number;
v_insert_count number;
v_update_count number;
v_result_num := WsWrkTask
( p_job_id, p_task_id, p_sequence
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
, v_insert_count, v_update_count, 0, 0, 0, 0, 0
);
```

DB2

DB2 Parameters: WsWrkTask

Callable Routine Type: PROCEDURE.

Parameter Name	Datatype	Mode
p_job_key	INTEGER	IN
p_task_key	INTEGER	IN
p_sequence	INTEGER	IN
p_inserted	INTEGER	IN
p_updated	INTEGER	IN
p_replaced	INTEGER	IN
p_deleted	INTEGER	IN
p_discarded	INTEGER	IN
p_rejected	INTEGER	IN
p_errored	INTEGER	IN

DB2 Examples: WsWrkTask

```
-- The p_ VARIABLES are normally PARAMETERS in a RED-generated Procedure.
DECLARE p_sequence integer;
DECLARE p_job_name varchar(256);
DECLARE p_task_name varchar(256);
DECLARE p_job_id integer;
DECLARE p_task_id integer;
DECLARE p_return_msg varchar(256);
DECLARE p_status integer;
DECLARE v_insert_count integer;
DECLARE v_update_count integer;
CALL [METABASE].WsWrkTask
( p_job_id, p_task_id, p_sequence
, v_insert_count, v_update_count, 0, 0, 0, 0, 0
);
```

© Copyright, WhereScape Software Ltd All rights reserved.

Ws_admin_v Views

Admin views provide a means of interacting with the WhereScape RED metadata from within your chosen reporting tools.

The following admin views are available:

Name	Description
ws_admin_v_audit	Created from the ws_wrk_audit_log table
ws_admin_v_dim_col	Created from the ws_dim_tab and ws_dim_col tables
ws_admin_v_error	Created from the ws_wrk_error_log table
ws_admin_v_fact_col	Created from the ws_fact_tab and ws_fact_col tables
ws_admin_v_fact_join	Created from the ws_fact_tab and ws_fact_col tables
ws_admin_v_sched	Created from the ws_wrk_job_log table
ws_admin_v_task	Created from the ws_wrk_task_run and ws_wrk_task_log tables

Ws_admin_v_audit

This Audit view is created using columns from the ws_wrk_audit_log table.

Columns

The following columns are created:

Column	Description
wa_time_stamp	The date or time at which this view was created
wa_sequence	Refer to **Callable Routines Common Input** for details.
wa_job	Refer to **Callable Routines Common Input** for details.
wa_task	Refer to **Callable Routines Common Input** for details.
wa_status	Refer to **Callable Routines Common Input** for details.
wa_message	The message associated with this audit log
wa_db_msg_desc	The database message associated with this audit log

SQL Script

SQLQuery3.sql - l...TTLE-B\User (53))

```
USE [Warehouse]
GO

/****** Object:  View [dbo].[ws_admin_v_audit]    Script Date: 03/23/2
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

create view [dbo].[ws_admin_v_audit] (
wa_time_stamp, wa_sequence,wa_job, wa_task, wa_status,
wa_message,wa_db_msg_desc )
as
select  wa_time_stamp, wa_sequence,wa_job, wa_task, wa_status,
wa_message,wa_db_msg_desc
from ws_wrk_audit_log
--order by isnull(wa_row_number,0),wa_time_stamp, wa_job, wa_task
GO
```

© Copyright, WhereScape Software Ltd All rights reserved.

Ws_admin_v_dim_col

This Dimension Column view is created from the ws_dim_tab and ws_dim_col tables.

Columns

The following columns are created:

Column	Description
dimension_table	name of the dimension table
dim_display_value	display value of the dimension
dim_description	description of the dimension
dim_type	a character value indicating the dimension type • T - time dimension • V - dimension view • D - dimension • C - slowly changing dimension • P - previous value dimension • else - unknown
column_name	column name
business_name	business name chosen for the column
description	description of the dimension column
data_type	data type of the dimension column
source_table	source table of the dimension table
source_column	source column of the dimension column
nulls_allowed	flag indicating whether or not nulls are allowed
numeric	flag indicating whether or not the column is numeric
additive	flag indicating whether or not the column is additive
attribute	flag indicating whether or not the column is an attribute
format	the dimension column format
column_order	the dimension column order

© Copyright, WhereScape Software Ltd All rights reserved.

SQL Script

SQLQuery4.sql - I...TTLE-B\User (53))

```
create view [dbo].[ws_admin_v_dim_col] (
dimension_table,
dim_display_value,
dim_description,
dim_type,
column_name,
business_name,
description,
data_Type,
source_table,
source_column,
nulls_allowed,
numeric,
additive,
attribute,Format,column_order)
AS
select
dt_table_name,
dt_display_name,
dt_description,
CASE dt_type_ind
   WHEN 'T' THEN 'Time dimension'
   WHEN 'V' THEN 'Dimension View'
   WHEN 'D' THEN 'Dimension'
   WHEN 'C' THEN 'Slowly changing dimension'
   WHEN 'P' THEN 'Previous value dimension'
   ELSE 'Unknown'
END,
dc_col_name ,
dc_display_name,
dc_src_strategy,
dc_data_type,
dc_src_table,
dc_src_column,
dc_nulls_flag,
dc_numeric_flag,
dc_additive_flag,
dc_attribute_flag,
dc_format,
dc_order
from ws_dim_tab LEFT OUTER JOIN ws_dim_col
on dt_obj_key = dc_obj_key
where dt_type_ind in ('D','C','T','V','P')
and isnull(dc_eul_flag,'N') = 'Y'
GO
```

© Copyright, WhereScape Software Ltd All rights reserved.

Ws_admin_v_error

This Error view is created using columns from the ws_wrk_error_log table.

Columns

The following columns are created:

Column	Description
wd_time_stamp	The date or time at which this view was created
wd_sequence	Refer to **Callable Routines Common Input** for details.
wd_job	Refer to **Callable Routines Common Input** for details.
wd_task	Refer to **Callable Routines Common Input** for details.
wd_status	Refer to **Callable Routines Common Input** for details.
wd_message	the message associated with this audit log
wd_db_msg_desc	the database message associated with this audit log

SQL Script

SQLQuery5.sql - I...TTLE-B\User (53))

```
USE [Warehouse]
GO

/****** Object:  View [dbo].[ws_admin_v_error]    Script Date: 03/23/2012 11:56:46 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

create view [dbo].[ws_admin_v_error] ( wd_time_stamp, wd_sequence,wd_job, wd_task, wd_status,
wd_message,wd_db_msg_desc )
as
select  wd_time_stamp, wd_sequence,wd_job, wd_task, wd_status,
wd_message,wd_db_msg_desc
from ws_wrk_error_log
--order by isnull(wd_row_number,0),wd_time_stamp, wd_job, wd_task
GO
```

© Copyright, WhereScape Software Ltd All rights reserved.

Ws_admin_v_fact_col

This Fact Column view is created from the ws_fact_tab and ws_fact_col tables.

Columns

The following columns are created:

Column	Description
fact_table	name of the fact table
fact_display_value	display value of the fact table
fact_description	description of the fact table
fact_type	a character value indicating the fact type • D - detail • R - rollup / combined • S - kpi • P - detail partitioned • Q - rollup partitioned • else unknown
column_name	column name
business_name	business name chosen for the column
description	description of the fact column
data_type	data type of the fact column
source_table	source table of the fact table
source_column	source column of the dimension column
nulls_allowed	flag indicating whether or not nulls are allowed
numeric	flag indicating whether or not the column is numeric
additive	flag indicating whether or not the column is additive
attribute	flag indicating whether or not the column is an attribute
format	the fact column format
column_order	the fact column order

© Copyright, WhereScape Software Ltd All rights reserved.

SQL Script

SQLQuery6.sql - I...TTLE-B\User (53))

```
create view [dbo].[ws_admin_v_fact_col] (
"fact_table",
"fact_display_value",
"fact_description",
"fact_type",
"column_name",
"business_name",
"description",
"data_Type",
"source_table",
"source_column",
"nulls_allowed",
"numeric",
"additive",
"attribute",
"format",
"column_order")
as
select
ft_table_name,
ft_display_name,
ft_description,
CASE ft_type_ind
   WHEN 'D' THEN 'Detail'
   WHEN 'R' THEN 'Rollup/Combined'
   WHEN 'S' THEN 'KPI'
   WHEN 'P' THEN 'Detail partitioned'
   WHEN 'Q' THEN 'Rollup partitioned'
   ELSE 'Unknown'
END,
fc_col_name,
fc_display_name,
fc_src_strategy,
fc_data_type,
fc_src_table,
fc_src_column,
fc_nulls_flag,
fc_numeric_flag,
fc_additive_flag,
fc_attribute_flag,
fc_format,
fc_order
from ws_fact_tab LEFT OUTER JOIN ws_fact_col
on ft_obj_key = fc_obj_key
where ft_type_ind in ('D','S','R','P','Q')
and isnull(fc_eul_flag,'N') = 'Y'
GO
```

© Copyright, WhereScape Software Ltd All rights reserved.

Ws_admin_v_fact_join

This Fact Column view is created from the ws_fact_tab and ws_fact_col tables.

Columns

The following columns are created:

Column	Description
fact_table	fact table name
fact_column	fact table column name
dimension_table	dimension table name
dimension_column	dimension table column name

SQL Script

SQLQuery7.sql - I...TTLE-B\User (53))

```
USE [Warehouse]
GO

/****** Object:  View [dbo].[ws_admin_v_fact_join]
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

create view [dbo].[ws_admin_v_fact_join] (
fact_table,
fact_column,
dimension_table,
dimension_column)
as
select
ft_table_name ,
fc_col_name,
fc_src_table,
fc_src_column
from ws_fact_tab LEFT OUTER JOIN ws_fact_col
on ft_obj_key = fc_obj_key
where ft_type_ind in ('D','S','R','Q','P')
and isnull(fc_join_flag,'N') = 'Y'
GO
```

Ws_admin_v_sched

This Scheduled Job view is created from the ws_wrk_job_log table.

Columns

The following columns are created:

Column	Description
type	"Waiting"
job_name	the name of the job

© Copyright, WhereScape Software Ltd All rights reserved.

Column	Description
status	a character value indicating the job status • H - on hold • R - running • P - pending • W - waiting • C - completed • B - blocked • F - failed • G - failed - aborted • E - error completion • else unknown
sequence	sequence number of the job
started_scheduled	
completed	date completed
hours_elapsed	hours elapsed since job started
minutes_elapsed	minutes elapsed since job started
okay	okay count
info	info count
warn	warning count
detail	detail count
error	error count

SQL Script

© Copyright, WhereScape Software Ltd All rights reserved.

SQLQuery8.sql - l...TTLE-B\User (53))

```
USE [Warehouse]
GO

/****** Object:  View [dbo].[ws_admin_v_sched]    Script
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

create view [dbo].[ws_admin_v_sched] (
    "type",
    "job_name",
    "status",
    "sequence",
    "started_scheduled",
    "completed",
    "hours_elapsed" ,
    "minutes_elapsed",
    "okay",
    "info",
    "warn",
    "detail",
    "error")
as
select
    'Waiting',
    wjc_name,
        CASE wjc_status
           WHEN 'H' THEN 'On Hold'
           WHEN 'R' THEN 'Running'
           WHEN 'P' THEN 'Pending'
           WHEN 'W' THEN 'Waiting'
           WHEN 'C' THEN 'Completed'
           WHEN 'B' THEN 'Blocked'
           WHEN 'F' THEN 'Failed'
           WHEN 'G' THEN 'Failed - Aborted'
           WHEN 'E' THEN 'Error Completion'
           ELSE 'Unknown'
        END,
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
    wjc_sequence,
    wjc_start_after,
    GETDATE(),
    0 ,
    0,
    0,
    0,
    0,
    0,
    0
      from ws_wrk_job_ctrl
union
     select 'Runnning',wjr_name,
        CASE wjr_status
           WHEN 'H' THEN 'On Hold'
           WHEN 'R' THEN 'Running'
           WHEN 'P' THEN 'Pending'
           WHEN 'W' THEN 'Waiting'
           WHEN 'C' THEN 'Completed'
           WHEN 'B' THEN 'Blocked'
           WHEN 'F' THEN 'Failed'
           WHEN 'G' THEN 'Failed - Aborted'
           WHEN 'E' THEN 'Error Completion'
           ELSE 'Unknown'
        END,
    wjr_sequence, wjr_started, wjr_completed,
    CONVERT(integer,DATEDIFF(mi,wjr_started,GETDATE())/60),
    DATEDIFF(mi,wjr_started,GETDATE()) -
    CONVERT(integer,DATEDIFF(mi,wjr_started,GETDATE())/60)*60,
     wjr_okay_count,
     wjr_info_count, wjr_warning_count, wjr_detail_count,
     wjr_error_count
      from ws_wrk_job_run
union
     select 'Finished',wjl_name,
        CASE wjl_status
           WHEN 'H' THEN 'On Hold'
           WHEN 'R' THEN 'Running'
           WHEN 'P' THEN 'Pending'
           WHEN 'W' THEN 'Waiting'
           WHEN 'C' THEN 'Completed'
           WHEN 'B' THEN 'Blocked'
           WHEN 'F' THEN 'Failed'
           WHEN 'G' THEN 'Failed - Aborted'
           WHEN 'E' THEN 'Error Completion'
           ELSE 'Unknown'
        END,
     wjl_sequence, wjl_started, wjl_completed,
     wjl_elapsed_hh, wjl_elapsed_mi, wjl_okay_count,
     wjl_info_count, wjl_warning_count, wjl_detail_count,
     wjl_error_count
     from ws_wrk_job_log
GO
```

© Copyright, WhereScape Software Ltd All rights reserved.

Ws_admin_v_task

This Task view is created from the ws_wrk_task_run and ws_wrk_task_log tables.

Columns

The following columns are created:

Column	Description
result	
task_name	the name of the task
status	a character value indicating the task status • H - on hold • R - running • P - pending • W - waiting • C - completed • B - blocked • F - failed • G - failed - aborted • E - error completion • else unknown
sequence	sequence number of the task
started	
completed	
hours_elapsed	hours elapsed since the task started
minutes_elapsed	minutes elapsed since the task started
info	info count
warn	warning count
detail	detail count
inserted	record inserted
updated	record updated
replaced	record replaced
deleted	record deleted
discarded	record discarded
rejected	record rejected
errored	record errored

SQL Script

© Copyright, WhereScape Software Ltd All rights reserved.

SQLQuery10.sql -...TTLE-B\User (53))

```
USE [Warehouse]
GO

/****** Object:  View [dbo].[ws_admin_v_task]    Script Date
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

create view [dbo].[ws_admin_v_task] (
    "Result",
    "task_name",
    "status",
    "sequence",
    "started",
    "completed",
    "hours_elapsed" ,
    "minutes_elapsed",
    "info",
    "warn",
    "detail",
    "inserted",
    "updated",
    "replaced",
    "deleted",
    "discarded",
    "rejected",
    "errored")
as
    Select
    'Runnning',wtr_name,
        CASE wtr_run_status
           WHEN 'H' THEN 'On Hold'
           WHEN 'R' THEN 'Running'
           WHEN 'P' THEN 'Pending'
           WHEN 'W' THEN 'Waiting'
           WHEN 'C' THEN 'Completed'
           WHEN 'B' THEN 'Blocked'
           WHEN 'F' THEN 'Failed'
           WHEN 'G' THEN 'Failed - Aborted'
           WHEN 'E' THEN 'Error Completion'
           ELSE 'Unknown'
        END,
```

© Copyright, WhereScape Software Ltd All rights reserved.

```
    wtr_sequence, wtr_started, wtr_completed,
    CONVERT(integer,DATEDIFF(mi,wtr_started,GETDATE())/60),
    DATEDIFF(mi,wtr_started,GETDATE()) -
    CONVERT(integer,DATEDIFF(mi,wtr_started,GETDATE())/60)*60,
      wtr_info_count, wtr_warning_count, wtr_detail_count,
     wtr_rec_inserted, wtr_rec_updated,wtr_rec_replaced,wtr_rec_deleted,wtr_rec_discarded,
    wtr_rec_rejected,wtr_rec_errored
      from ws_wrk_task_run
union
     select wtl_return_msg,wtl_name,
        CASE wtl_run_status
           WHEN 'H' THEN 'On Hold'
           WHEN 'R' THEN 'Running'
           WHEN 'P' THEN 'Pending'
           WHEN 'W' THEN 'Waiting'
           WHEN 'C' THEN 'Completed'
           WHEN 'B' THEN 'Blocked'
           WHEN 'F' THEN 'Failed'
           WHEN 'G' THEN 'Failed - Aborted'
           WHEN 'E' THEN 'Error Completion'
           ELSE 'Unknown'
        END,
     wtl_sequence, wtl_started, wtl_completed,
     wtl_elapsed_hh, wtl_elapsed_mi,
     wtl_info_count, wtl_warning_count, wtl_detail_count,
     wtl_rec_inserted, wtl_rec_updated,wtl_rec_replaced,wtl_rec_deleted,wtl_rec_discarded,
    wtl_rec_rejected,wtl_rec_errored
     from ws_wrk_task_log
GO
```

© Copyright, WhereScape Software Ltd All rights reserved.

Retrofitting

WhereScape RED includes an advanced retrofit capability that can be used to:

1. Migrate an existing data warehouse from one relational database to another (known as fork-lifting).
2. Load a data model from a modeling tool.

Retrofitting is achieved using the **Retro** object type in WhereScape RED and the **Retrofit tables** wizard.

For information on migrating an existing data warehouse, see Migrating the Data Warehouse Database Platform (**OLAP Retrofitting an OLAP Object**, **Migrating the Data Warehouse Database Platform**).

For information on importing a data model, refer to **Importing a Data Model**.

Migrating the Data Warehouse Database Platform

WhereScape RED has an advanced retrofitting wizard for migrating an existing data warehouse from one relational database to another.

The process to migrate an existing data warehouse includes:

1. Creating a connection object to the existing warehouse database.
2. Creating Retro objects based on the source tables in the existing warehouse database.
3. Setting the Retro objects as Retro Copy type objects.
4. Running a Scheduler task to build the Retro Copy objects from the source tables.
5. Setting the Retro objects back to Retro (Retro Definition) type objects.
6. Converting the Retro objects to the Target Object types.

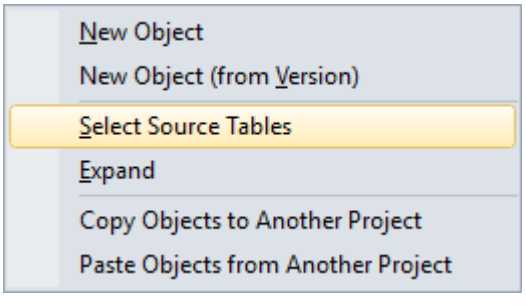
The steps to use this wizard are as follows:

1. Create a connection object for the old data warehouse database, populating the following fields:
   - Connection Name
   - Connection Type => **ODBC**
   - ODBC Source
   - Work Directory
   - Extract user name
   - Extract password

   Note

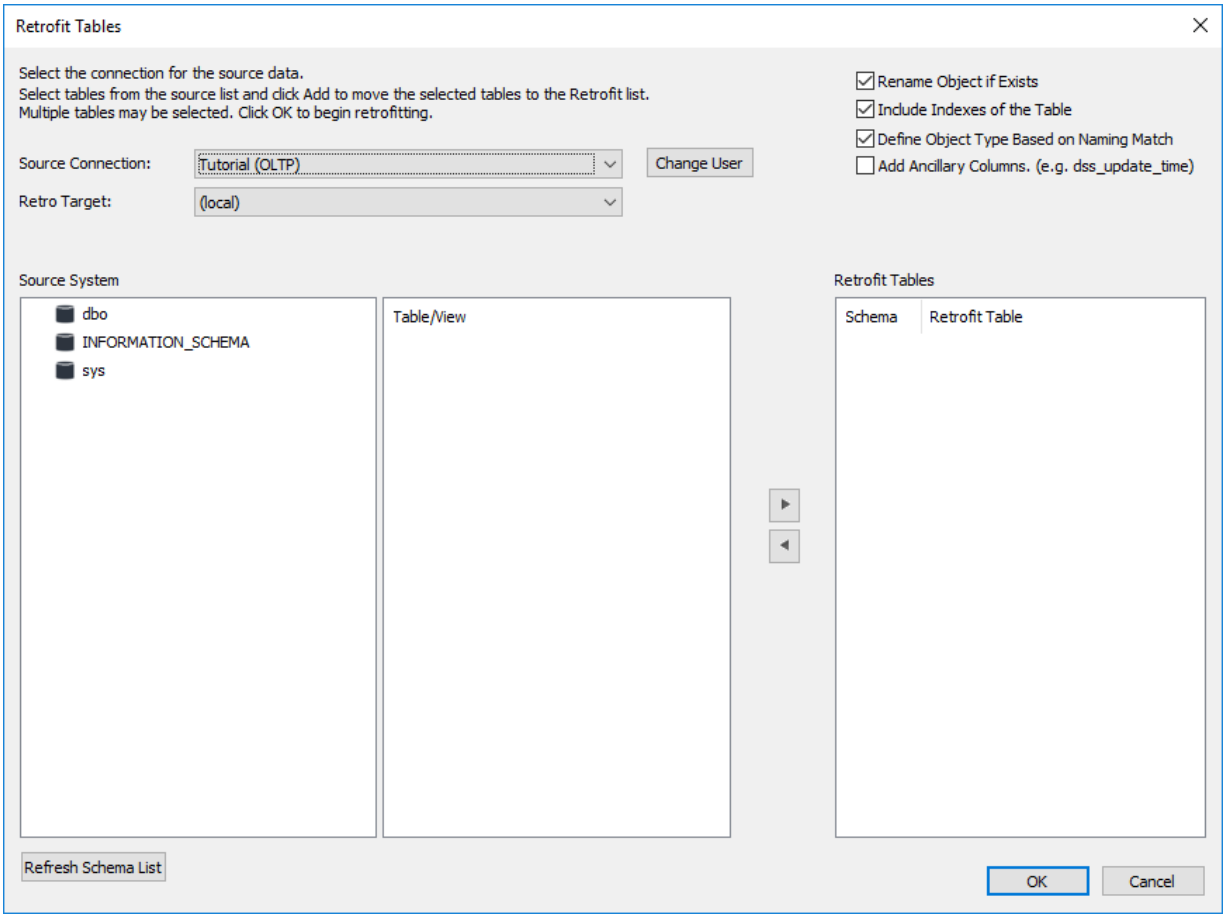
   The extract user must be able to select from all tables to be migrated.

2. Ensure all naming standards in **Tools > Options** are set to match the objects being retrofitted. This saves work later.
3. Ensure **Enable Retro** is selected in the **Tools > Options > Object Types** menu.
4. Right-click the **Retro object group** in the object tree in the left pane and then click **Select Source Tables**.



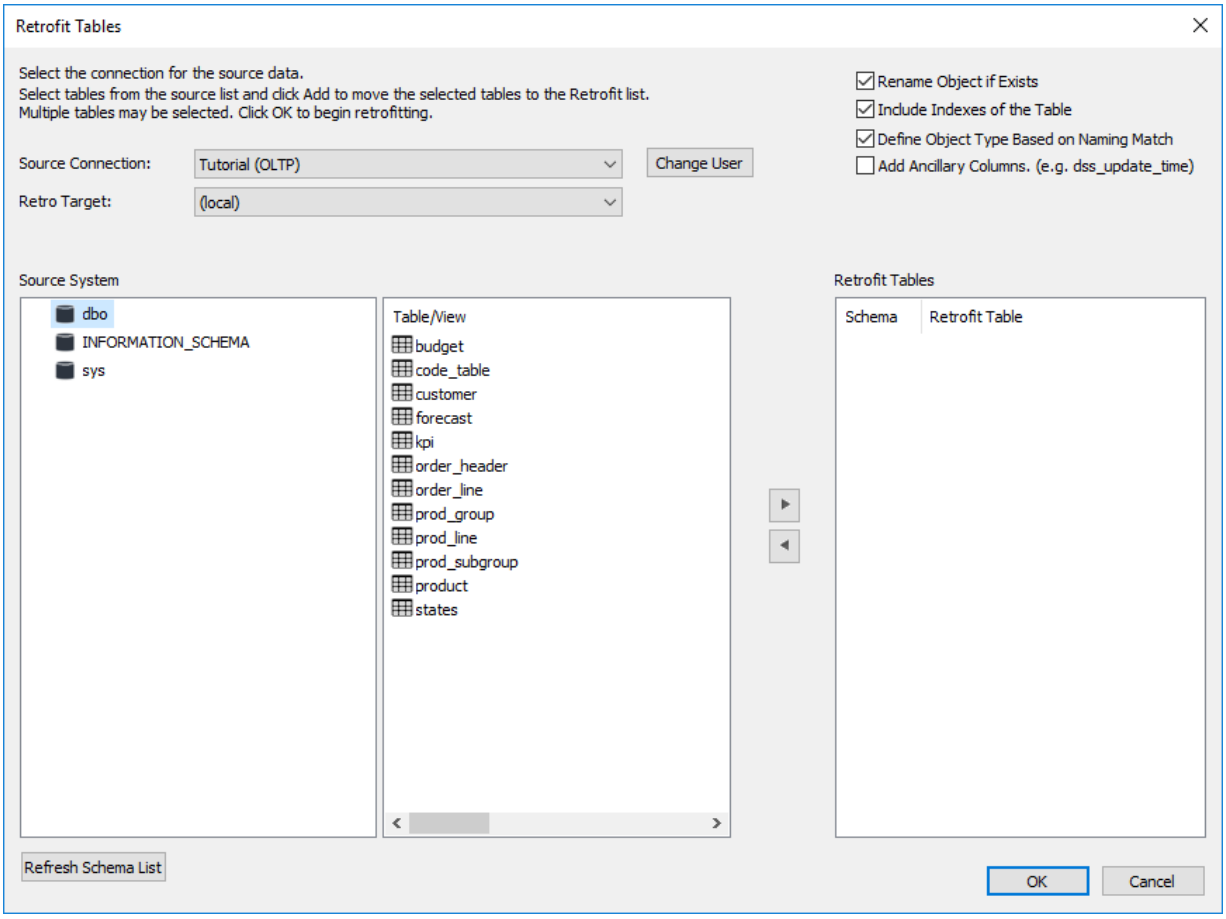
© Copyright, WhereScape Software Ltd All rights reserved.

5. The **Retrofit Tables** window appears. In the **Source Connection** drop-down list, choose the connection set up in step 1. A list of **databases** appears in the left pane.



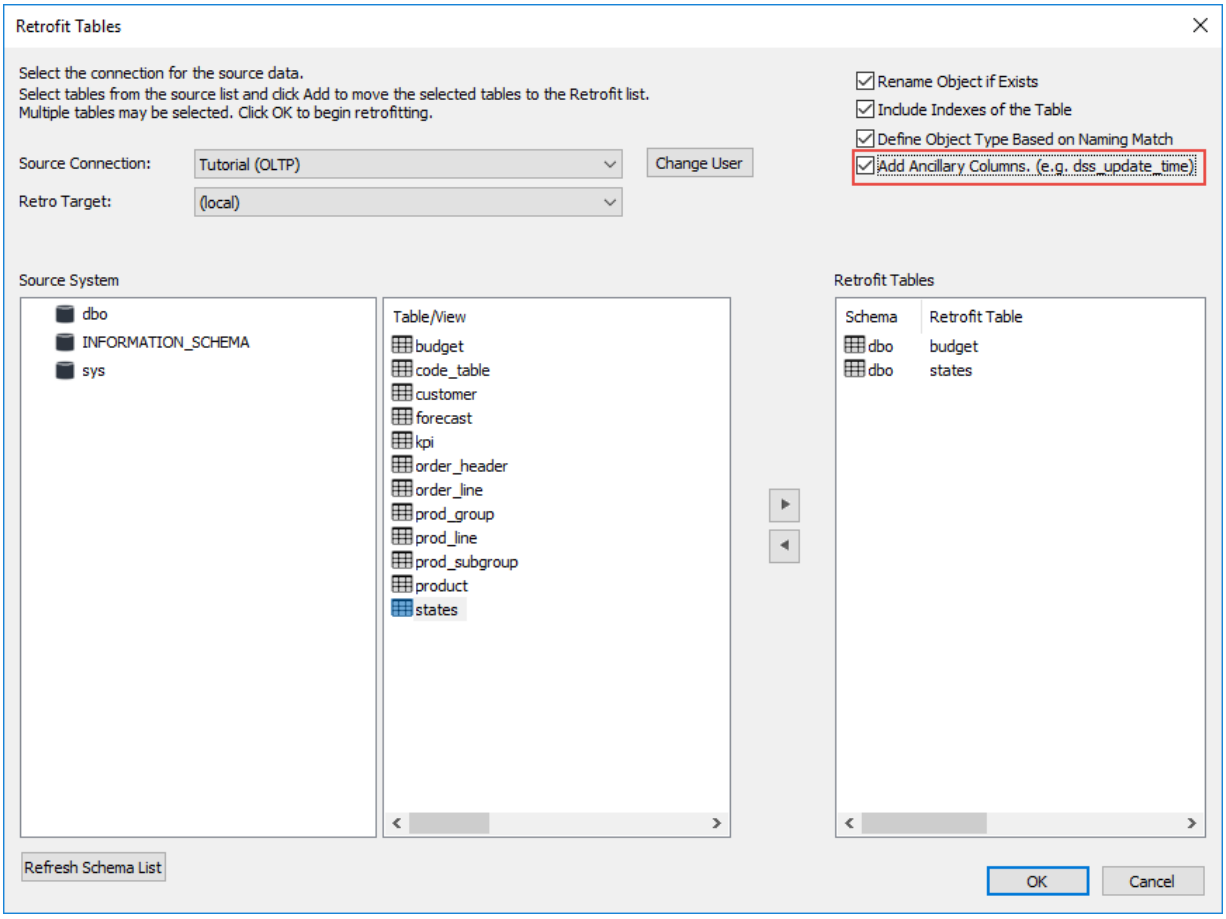
© Copyright, WhereScape Software Ltd All rights reserved.

6. Double-click the database/user/schema in the left pane list. A list of **tables** in the database is displayed in the middle pane.

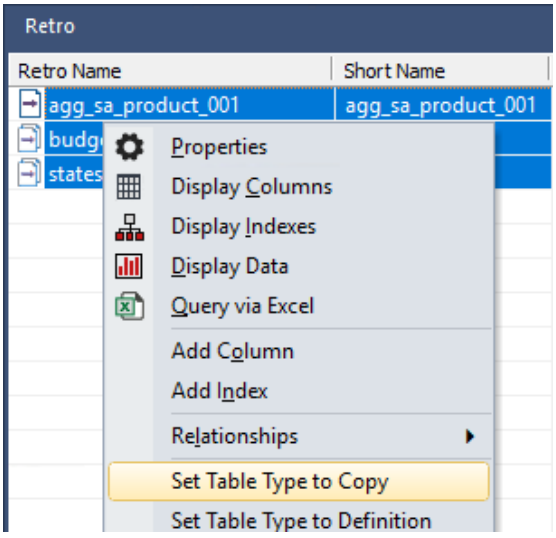


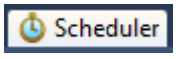
© Copyright, WhereScape Software Ltd All rights reserved.

7. Select all the required tables from the middle pane list and click > to move them to the right pane. Then click the **Add Ancillary Columns (e.g. dss_update_time)** check-box and click **OK**.



8. WhereScape RED acquires the metadata for the tables being migrated and creates a new WhereScape RED Retro object for each.
9. Double-click the **Retro object group** in the left pane. Select all Retros in the middle pane. Right-click and select **Set Table Type to Copy**. This enables the data in the legacy data warehouse to be copied across to the new data warehouse.



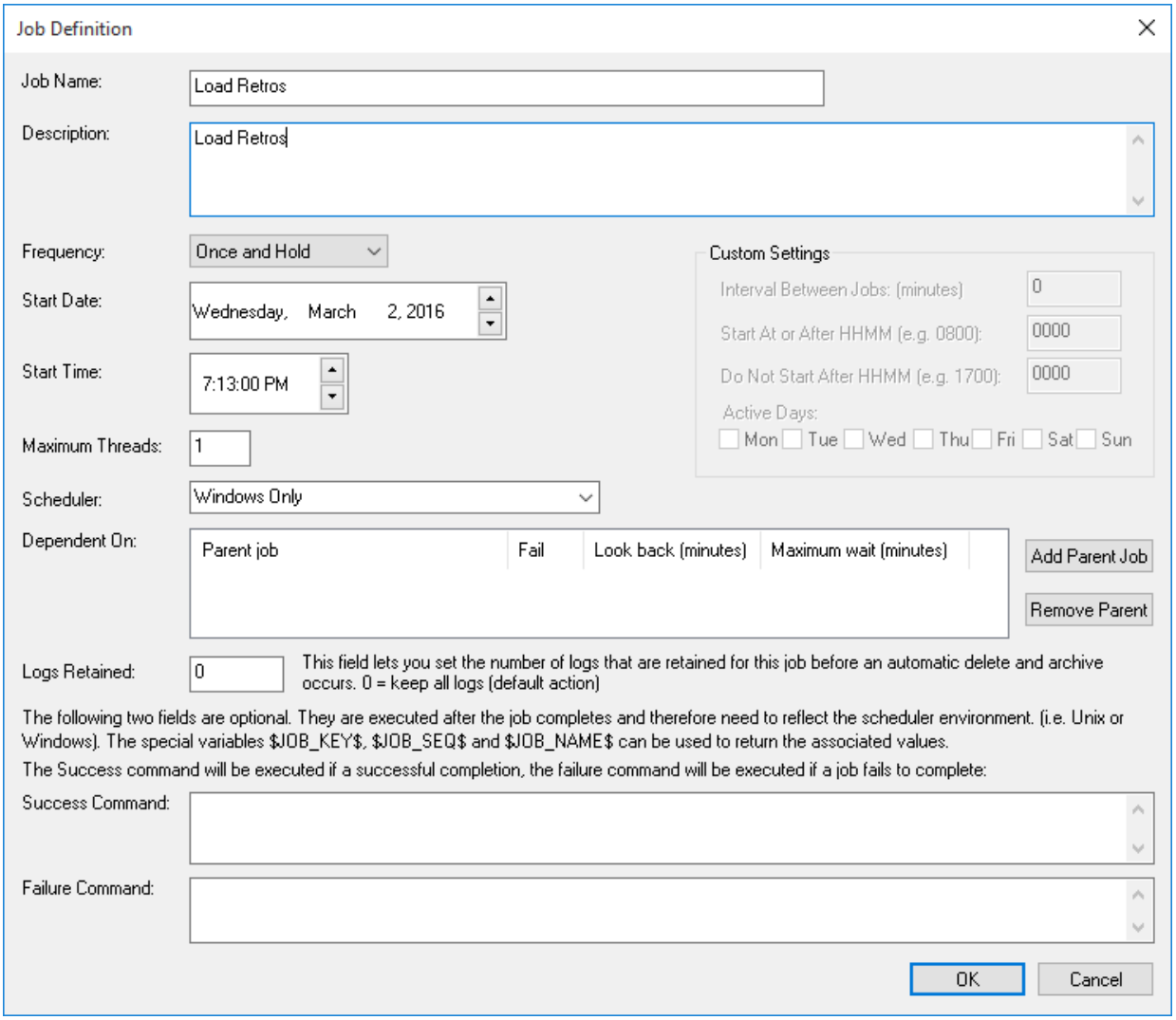
10. Click the **Scheduler** button from the toolbar. 

© Copyright, WhereScape Software Ltd All rights reserved.

11. Create a new job to run straight away and click **OK**.

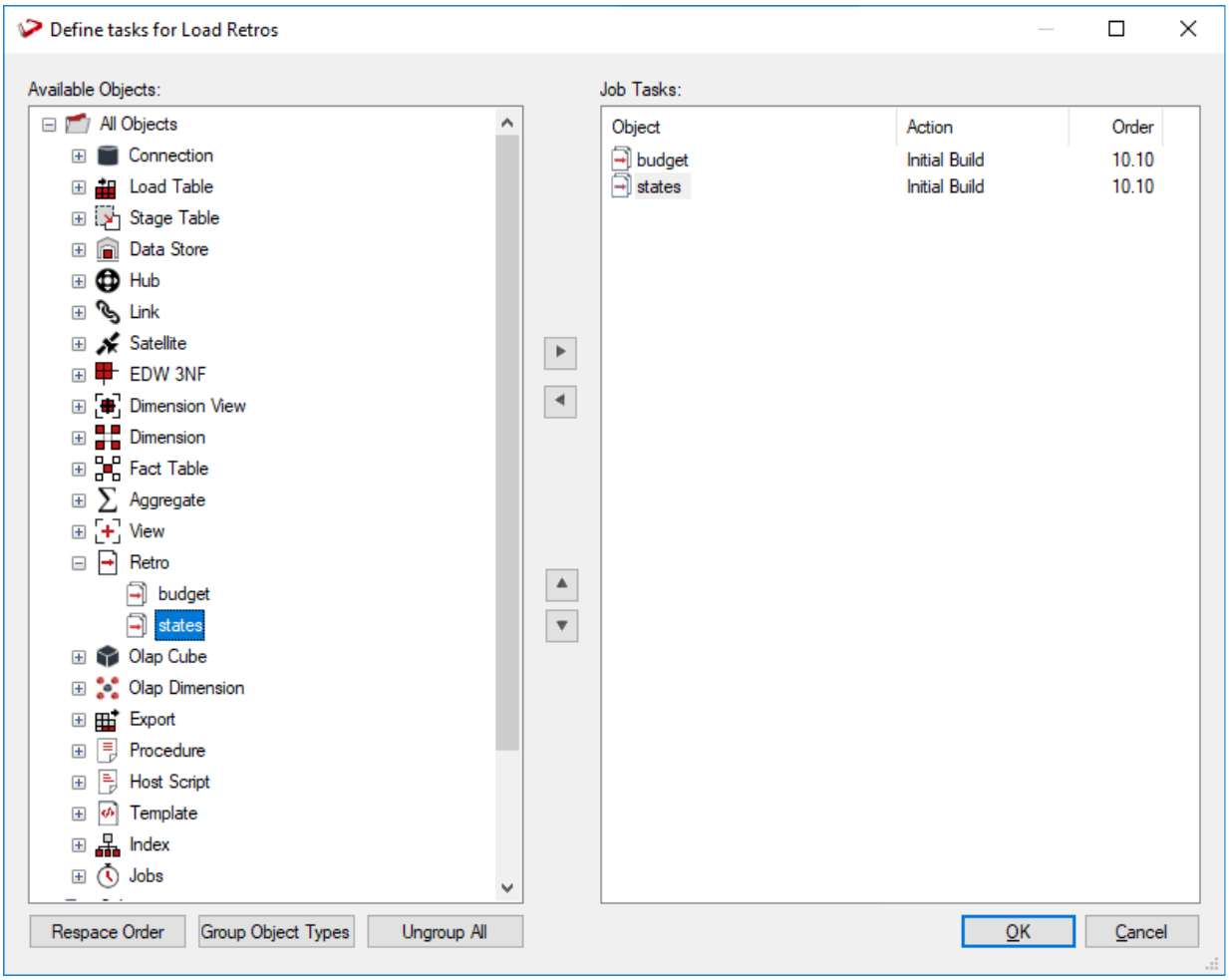
Note:

A scheduler must be running on the data warehouse connection for this job to complete, refer to the Scheduler Installation and Configuration section of the RED Installation Guide.

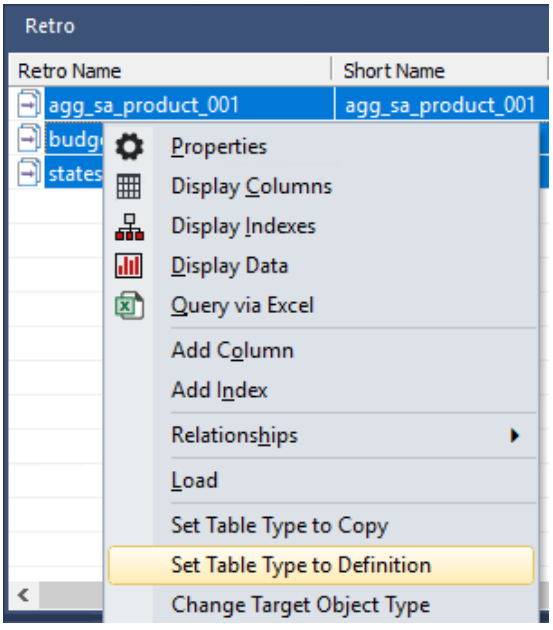


© Copyright, WhereScape Software Ltd All rights reserved.

12. Add all Retro objects created in steps (3) to (9) and then click **Group Object Types**.



13. Once the job is completed, return to the WhereScape RED Builder window. Double-click the **Retro object group**. Select all objects in the middle pane and then from the right-click context menu select **Set Table Type to Definition**. This indicates the data has been copied into the Retro objects and the Retros can then be converted to the target objects.

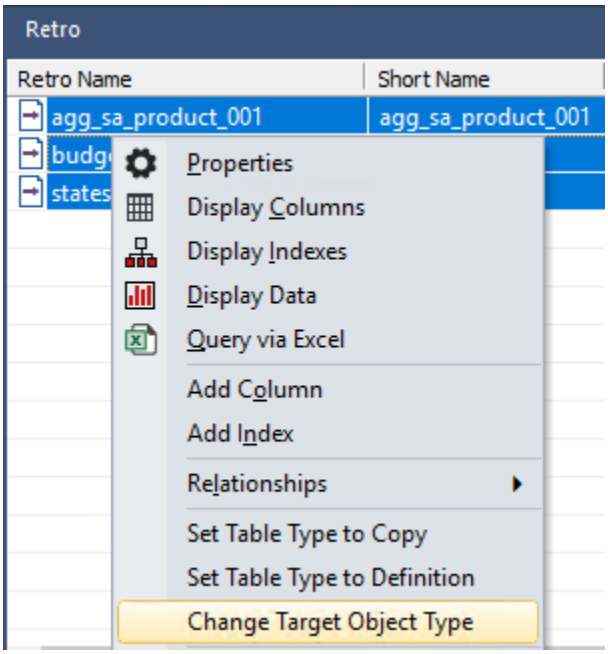


© Copyright, WhereScape Software Ltd All rights reserved.

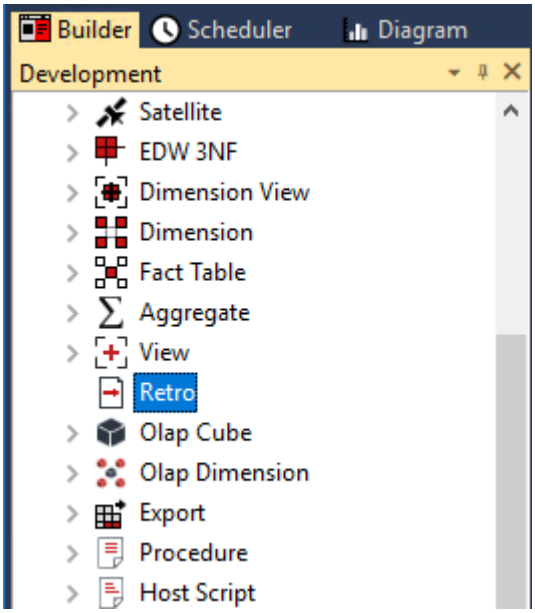
14. In the middle pane, select all objects. Right-click and select **Convert to Target Object**. WhereScape RED converts the **Retro** objects to the appropriate object types.

Note:

If the appropriate Target Object Type has not been set for one or more Retro objects; from the right click context menu select **Change Target Object Type** and select the correct Object Type.



15. There are no longer any Retro objects. They have been converted to Load, Stage, Dimension or Fact objects.



16. Change the source table and source column values on all of the retrofitted objects using either the Re-target source table window, or by editing column properties. Refer to **Re-Targeting Source Tables** for details.
17. Convert the old data warehouses code to WhereScape RED procedure in the new data warehouse database. Refer to **Integrate Procedures** for details.
18. If necessary, create new connections to be used with any migrated Lad tables. Attach a connection to each Load table. Refer to **Loading Data** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

Importing a Data Model

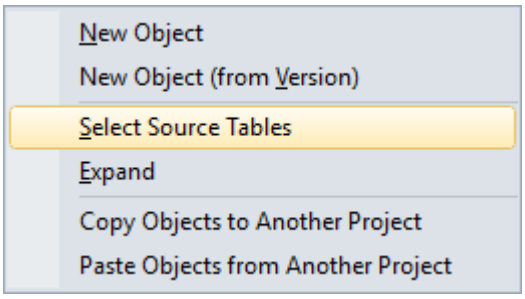
WhereScape RED provides functionality for importing data models from modeling tools.

The process to import a model is:

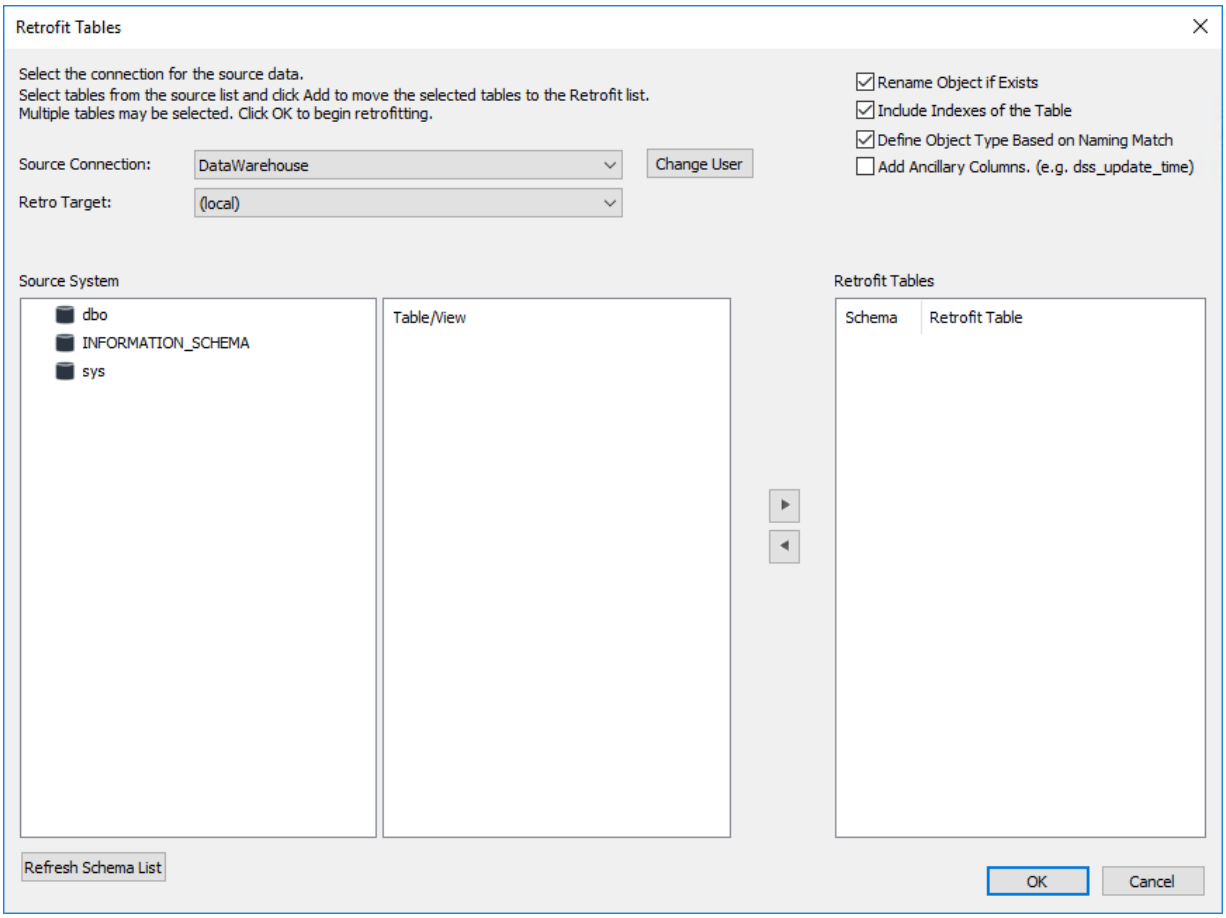
1. Create the physical data model in the modeling tool.
2. Generate DDL for the physical model in the modeling tool.
3. Run the DDL in the data warehouse database to create empty versions of the model tables.
4. Retrofit the tables in the dummy database into the WhereScape RED metadata as Retro objects.
5. Convert the Retro objects to Dimensions and Facts.

The following instructions outline steps 4 and 5 above:

1. Right-click the **Retro object group** in the object tree in the left pane and select **Select Source Tables**.

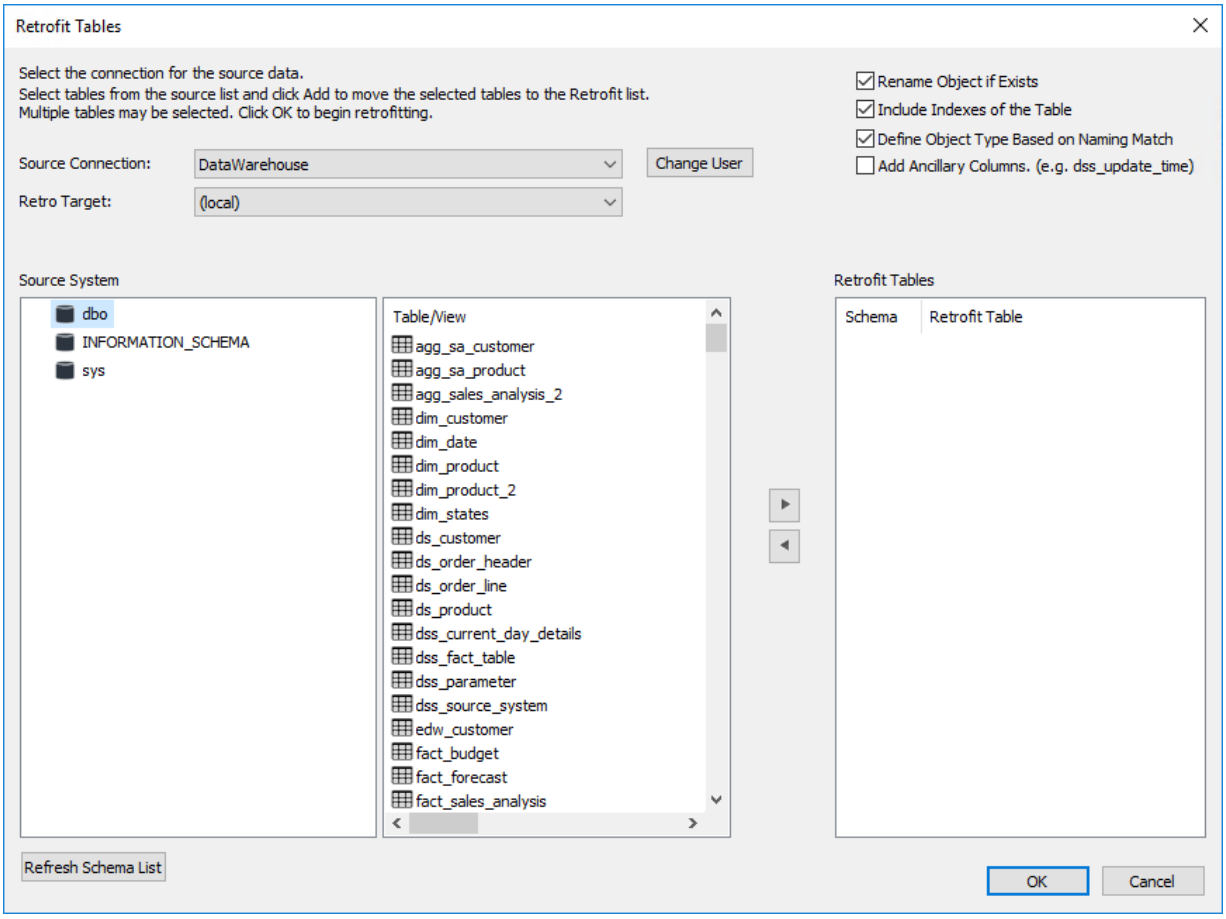


2. The **Retrofit tables** dialog is displayed. In the **Source Connection** drop-down list choose the **Data Warehouse** connection. A list of **databases** appears in the left pane (your list will be different).



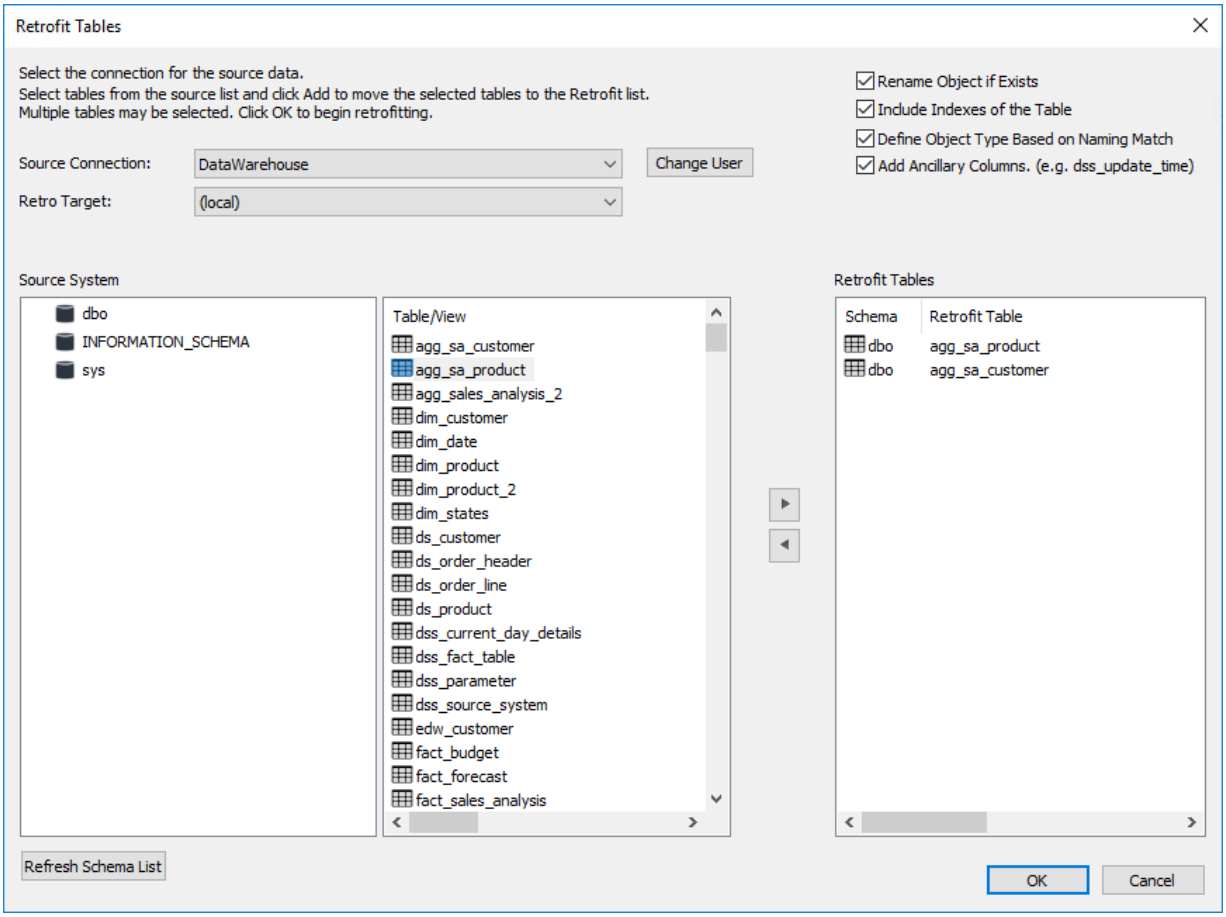
© Copyright, WhereScape Software Ltd All rights reserved.

3. Double-click the data warehouse database/schema in the left pane list. A list of **tables** in the database is displayed in the middle pane.



© Copyright, WhereScape Software Ltd All rights reserved.

4. Click the required tables in the middle pane list and click **>** to move them to the right pane. Then click the **Add Ancillary Columns (e.g. dss_update_time)** check box. Click **OK**.

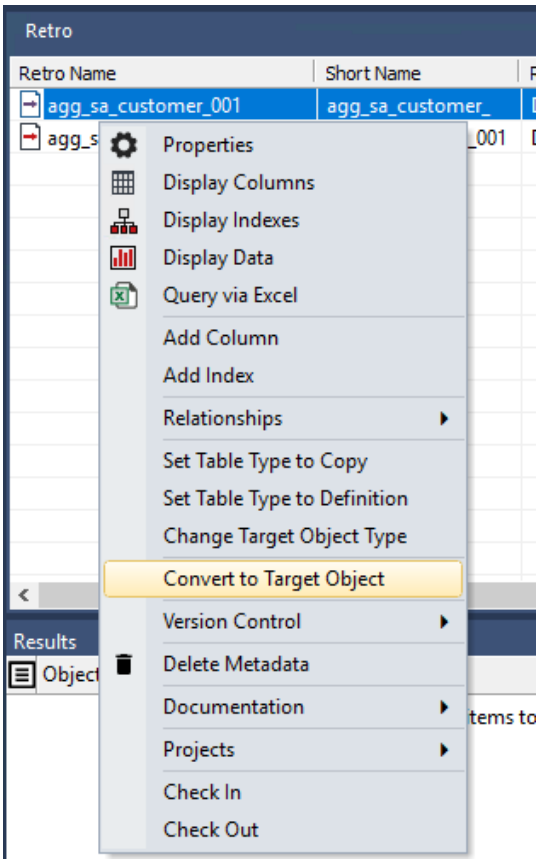


5. Double-click the **Retro object group** in the object tree in the left pane. Select all objects in the middle pane, right-click and then select **Convert to Target Object**.

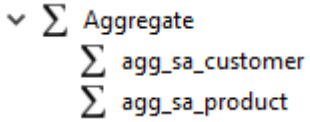
Note:

If the appropriate Target Object Type has not been set for one or more Retro objects; in the right click menu select **Change Target Object Type** and select the correct Object Type.

© Copyright, WhereScape Software Ltd All rights reserved.



6. The new aggregate tables are imported and listed in the **Aggregate** object group in the left pane.



7. At this stage, you have only created the table metadata. To create the tables in the data warehouse, double click the object group in the left pane. In the middle pane highlight the tables, then right-click and select **Create (ReCreate)**.

© Copyright, WhereScape Software Ltd All rights reserved.

Re-Targeting Source Tables

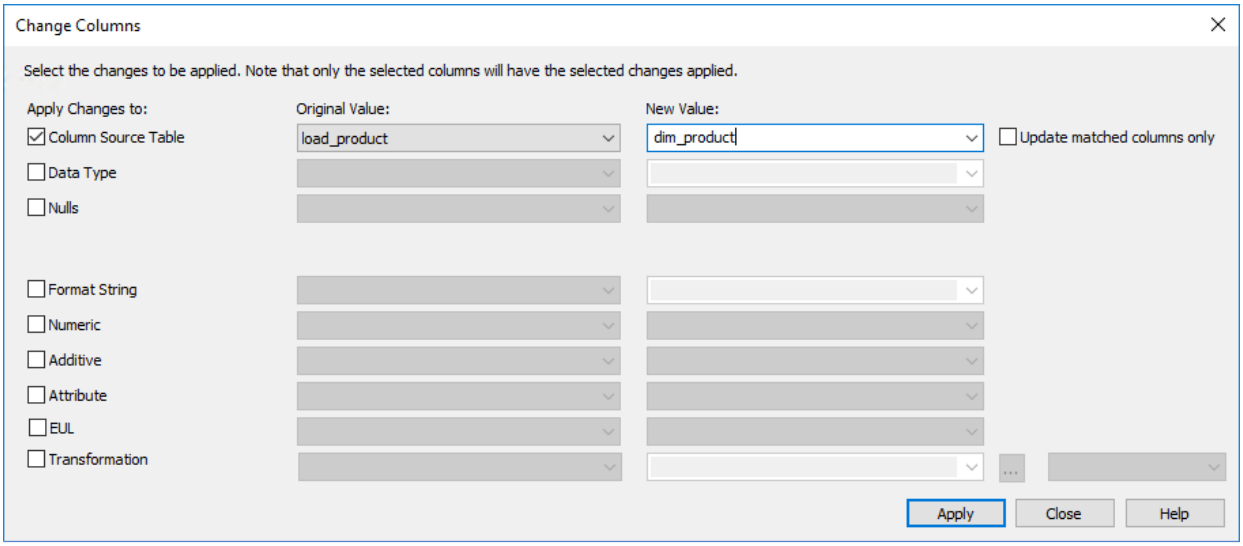
Objects that have been retrofitted into the WhereScape RED meta data have themselves as their source table:

dim_product Columns

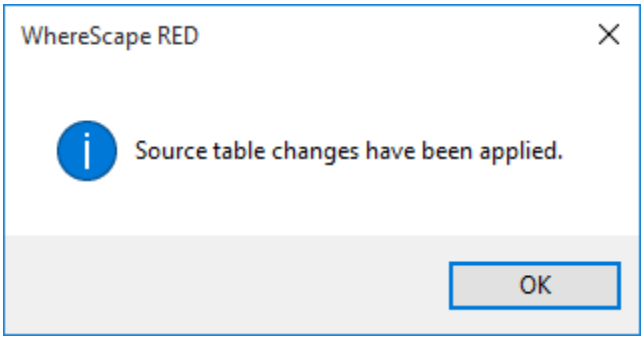
Column Name	Display Name	Data Type	Source Table	Source Column
dim_product_key	dim product key	integer ident...		dim_product_key
code	code	numeric(6)	load_product	code
description	description	varchar(64)	load_product	description
prod_line	prod line	varchar(24)	load_product	prod_line
prod_group	prod group	varchar(24)	load_product	prod_group
subgroup	subgroup	varchar(24)	load_product	subgroup
subgroup_description	subgroup desc...	varchar(64)	load_prod_subgro...	subgroup_descript...
line_description	line description	varchar(64)	load_prod_line	line_description
group_description	group descript...	varchar(64)	load_prod_group	group_description
dss_update_time	dss update time	datetime		dss_update_time

They can be re-targeted to the correct source table(s) using the WhereScape RED re-target wizard as follows.

1. Right-click a table object in the left pane and select **Change Column(s)**.
2. Select the **Column Source Table** check box. Select **load_product** from the **Original Value** drop-down list. Select **dim_product** from the **New Value** drop-down list and then click **Apply**.



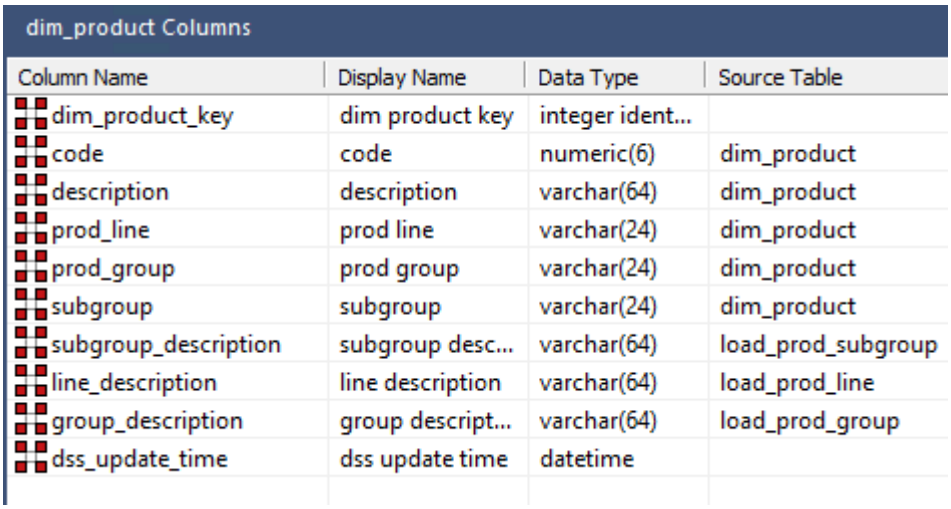
3. A message is displayed to show that the source columns have been changed, click **OK**.



4. Click **Close**.

© Copyright, WhereScape Software Ltd All rights reserved.

5. Confirm the **Source Table** column in the middle pane.



dim_product Columns

Column Name	Display Name	Data Type	Source Table
dim_product_key	dim product key	integer ident...	
code	code	numeric(6)	dim_product
description	description	varchar(64)	dim_product
prod_line	prod line	varchar(24)	dim_product
prod_group	prod group	varchar(24)	dim_product
subgroup	subgroup	varchar(24)	dim_product
subgroup_description	subgroup desc...	varchar(64)	load_prod_subgroup
line_description	line description	varchar(64)	load_prod_line
group_description	group descript...	varchar(64)	load_prod_group
dss_update_time	dss update time	datetime	

Retro Column Properties

Each Retro column has a set of associated properties. The definition of each property is defined below.

If the **Column name** or **Data type** is changed for a column then the metadata will differ from the table as recorded in the database. Use the **Validate > Validate Table Create Status** menu option to compare the metadata to the table in the database. When positioned on the table name after the validate has completed, a right-click menu option **Alter Table** will alter the database table to match the metadata definition.

Tip

If a database table's definition is changed in the metadata then the table will need to be altered in the database. Use the **Validate > Validate Table Create Status** to compare metadata definitions to physical database tables. The option also provides the ability to alter the database table through a pop-up menu option from the validated table name.

General

Fields	Description
Table Name	Database-compliant name of the table that contains the column. [Read-only].
Column Name	Database-compliant name of the column. Typically column-naming standards exclude the use of spaces etc. A good practice is to only use alphanumerics, and the underscore character. Changing this field alters the table's definition. **Note** A case conversion button on the right converts the text between different cases: UPPER CASE, Capitalized Case and lower case. The mode cycles to the next case in the sequence each conversion.
Business Display Name	Name that the business uses to refer to the column, which is included in the RED-generated documentation and can be used in the end user layer of other tools. [Does NOT affect the physical database table]. As such it is a free form entry and any characters are valid. **Note** A case conversion button on the right converts the text between different cases: UPPER CASE, Capitalized Case and lower case. The mode cycles to the next case in the sequence each conversion.
Column Description	This field contains the description for the column. In the case of dimension keys, this field is used to show the join between the Retro table and the dimension, once it has been defined as part of the Retro table update procedure generation.

© Copyright, WhereScape Software Ltd All rights reserved.

Physical Definition

Fields	Description
Column Order	Numeric value that controls the relative order of columns in the database create statement. The lowest numbered column will appear first in the table. Although this affects the physical table definition no action will be taken unless the table is re-created in the database. The columns can be re-numbered based on the existing order by choosing the **Respace Order Number** pop-up menu option when positioned over any column in the table. This action will number the columns in increments of 10 starting at 10. In addition to a simple change of the order field, the column order can be changed by first displaying the columns in the middle pane and then using drag and drop to move the columns around. This drag and drop process will automatically renumber the columns as required.
Data Type	Database-compliant data type that must be valid for the target database. Typical data types for Oracle are integer, number, char(), varchar2() and date. For SQL Server common types are integer, numeric, varchar() and datetime. Refer to the database documentation for a description of the data types available. Changing this field alters the table's definition.
Null Values Allowed	Determines whether the table column can hold NULL values or whether a value is always mandatory.
Default Value	Initial value that is assigned to the column when a row is inserted into the table but no value is specified for the column.

Meta Definition

Fields	Description
Format	Optional format mask that can be used in end user tools. [Does NOT affect the physical database table]. As such it is a free form entry and any characters are valid. Typically format masks are only used on numeric fields. Example: #,###0.00. It is not worth the effort of populating this field unless it can be utilized by the end user tools in use.
Numeric	Indicates whether the table column holds values that are numeric. This is normally only relevant for fact tables. It does not affect the physical table definition, but rather provides input to the view **ws_admin_v_dim_col** which can be used to assist in the population of an end user tool's end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.
Additive	Indicates whether the table column holds values that are additive. This implies that the column can be summed when performing data grouping in a query. This is normally only relevant for fact tables. It does not affect the physical table definition, but rather provides input to the view **ws_admin_v_dim_col** which can be used to assist in the population of an end user tool's end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.
Attribute	Indicates whether the table column holds values that are descriptive, and/or are used for grouping/summing. An attribute is defined as a column that is non factual. For example we may have an order number, or a invoice number stored in the fact table. Such columns are considered attributes, rather than facts. This checkbox is therefore normally only relevant for fact tables. This checkbox does not affect the physical table definition, but rather provides input to the view **ws_admin_v_dim_col** which can be used to assist in the population of an end user tools end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.
Business Key	Indicates whether the column is part of the BUSINESS key, which is defined during the update procedure build. [Normally maintained automatically]. Multiple columns can form the primary business key.
Key Type	Key type that is assigned and used when generating the table's update procedure and indexes. [Normally maintained automatically]. It can be altered here, but this should

© Copyright, WhereScape Software Ltd All rights reserved.

<table>
<tr><th>Fields</th><th>Description</th></tr>
<tr><td></td><td>only be done if the consequences are fully understood and tested.The supported values are:
<table>
<tr><th>Key type</th><th>Meaning</th></tr>
<tr><td>0</td><td>The artificial key. Set when the key is added during drag and drop table generation.</td></tr>
<tr><td>1</td><td>Component of all business keys. Indicates that this column is used as part of any business key. For example: By default the dss_source_system_key is added to every dimension table. It is considered part of any lookup on that table and has the key type set to 1. Set when the column is added during drag and drop table generation.</td></tr>
<tr><td>2</td><td>Indicates that this column is a dimensional join. Used on fact tables to indicate the dimension keys. Results in bitmap indexes being built for the columns. Set during the update procedure generation for a fact table, based on information from the staging table.</td></tr>
<tr><td>3</td><td>Slowly changing column indicator. Used on dimension tables to indicate that the column is being managed as a slowly changing column within the context of a slowly changing dimension. Set when a column is identified during the dimension update procedure generation.</td></tr>
<tr><td>4</td><td>Previous value column indicator. Used on dimension tables to indicate that the column is being managed as a previous value column. The source column identifies the parent column. Set during the dimension creation.</td></tr>
<tr><td>A</td><td>Indicates that the column is part of the primary business key. Set whenever a business key is defined as part of an update procedure generation.</td></tr>
<tr><td>B-Z</td><td>Indicates that the column is part of a secondary business key. Only used during index generation and not normally set.</td></tr>
</table></td></tr>
</table>

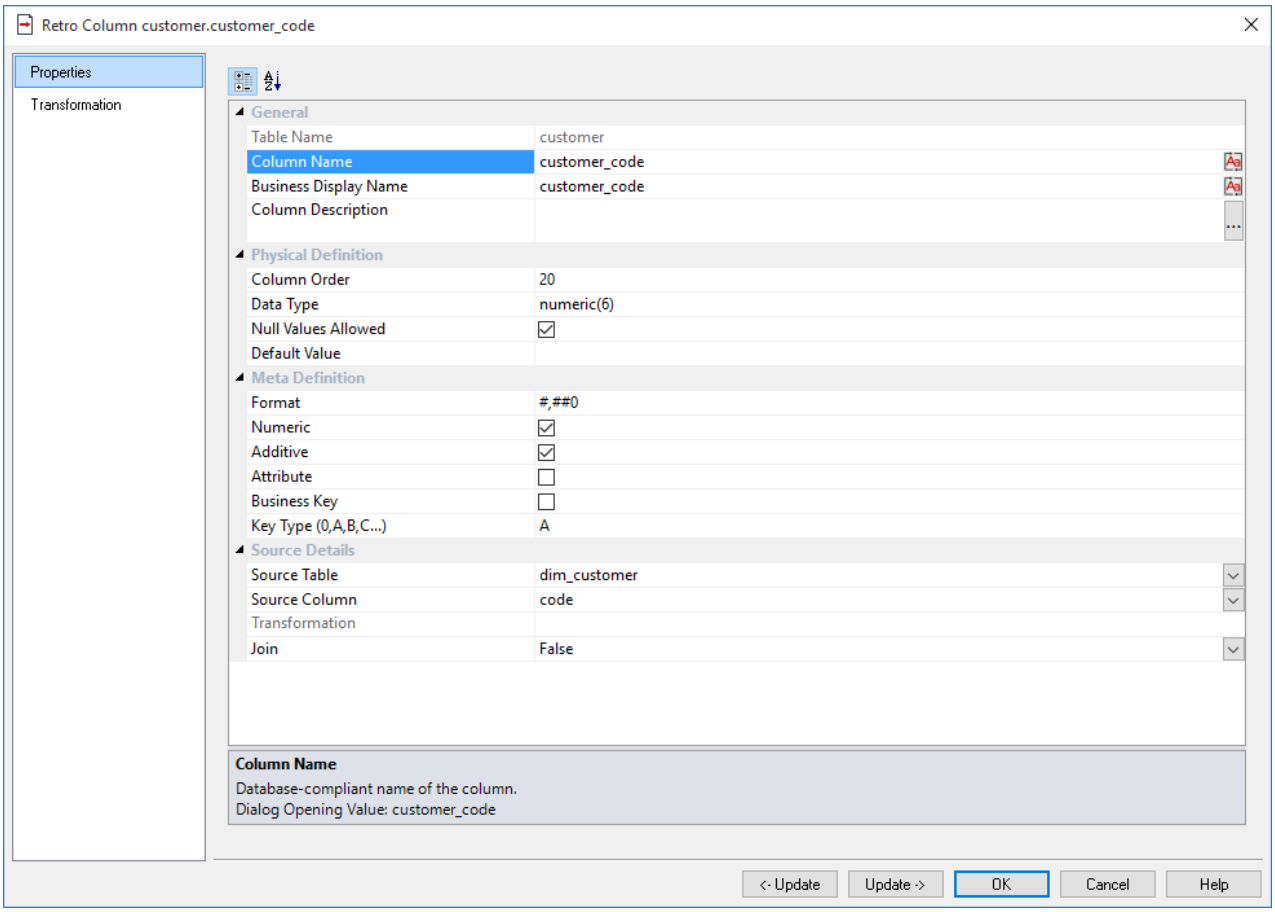
Source Details

Fields	Description
Source Table	Identifies the source table where the column's data comes from.
Source Column	Identifies the source column where the column's data comes from.
Transformation	Transformation. [Read-only].
Join	Indicates whether the table column is used in a table join. [Normally maintained automatically but can be optionally changed to override the default join logic used in the generated update procedure]. The **Source Table** and **Source Column** fields will provide the dimension table's side of the join. The options for this field are: False, True, Manual and Pre Join. • Setting this field to Manual changes the way the dimension table is looked up during the update procedure build. It allows you to join the dimension manually in the Cursor mapping dialog (where the 'Where' clause is built). • Setting this field to Pre Join activates the **Pre Join Source Table** field and allows you to select a table from the drop-down list.
Pre Join Source Table	Indicates the table from which the pre joined column was sourced. When the Join option is set to False, this field becomes inactive. When the Join option is set to True or Manual, this field is set to the current table name. When the Join option is set to Pre Join, then you can select the required table from the drop-down list.

© Copyright, WhereScape Software Ltd All rights reserved.

Retro Column Properties Screen

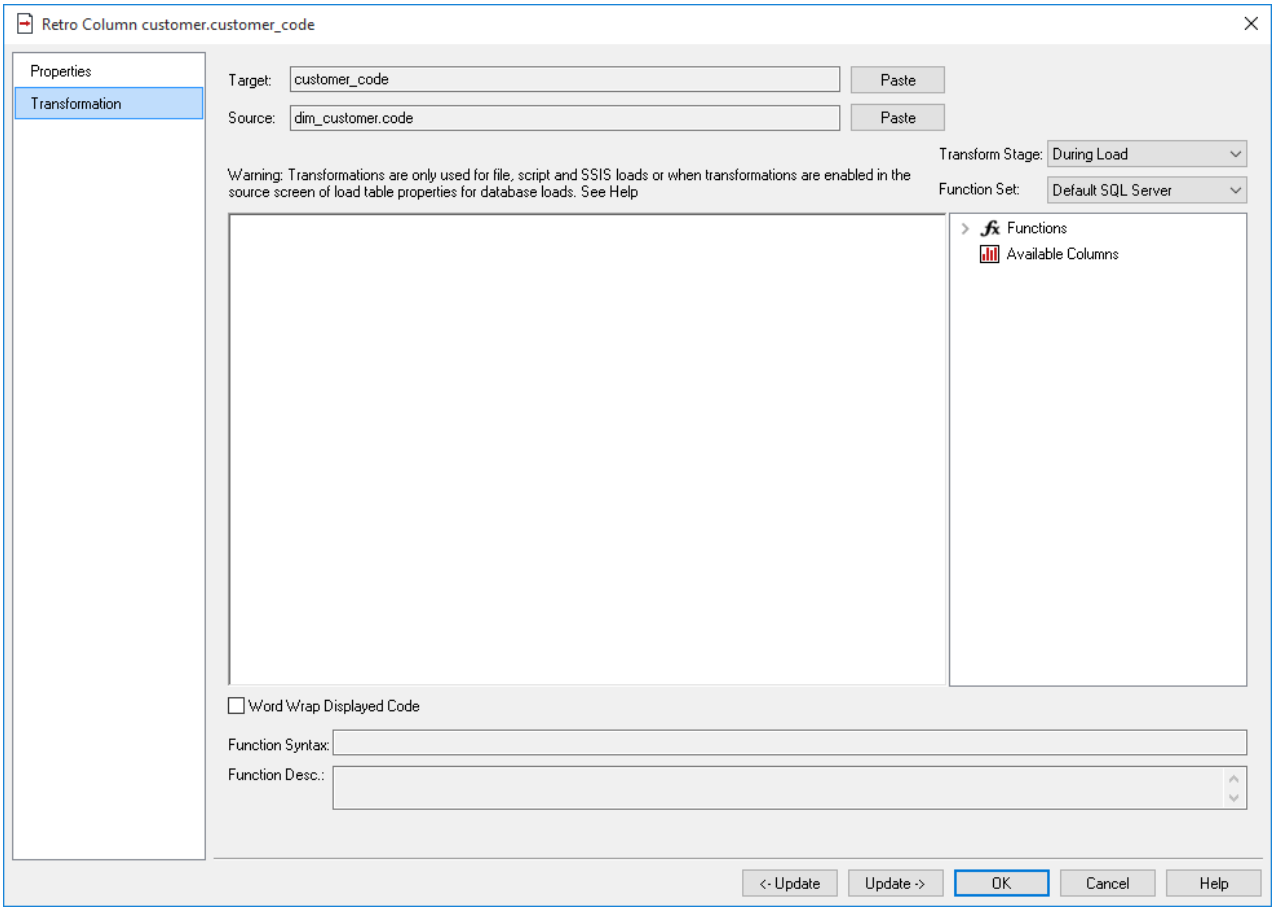
A sample **Properties** screen is as follows:



© Copyright, WhereScape Software Ltd All rights reserved.

Retro Column Transformations

It is possible to do transformations on Retro table columns. It is recommended that transformations are not performed on columns that are dimension keys or the business keys for the table. The transformation screen is as follows:



Note

Transformations are only put into effect when the procedure is re-generated.

Refer to **Transformations** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

Integrating WhereScape RED into an Existing Warehouse

Two main options exist in terms of bringing WhereScape RED into an existing data warehouse environment:

1. Rebuild tables and procedures with WhereScape RED.
2. Integrating existing tables and procedures into WhereScape RED.

Both options require manual coding changes to stored procedures. The main advantages and disadvantages of these two options are discusses below, and in detail in the following sections.

Rebuild

The rebuild option is essentially a redevelopment of the existing data warehouse utilizing the knowledge acquired in the initial development and the rapid development capabilities of WhereScape RED. A rebuild will take more time and effort than just integrating existing tables and procedures but will provide a better platform on which to extend the data warehouse. Refer to **Rebuilding** for details.

Integrate

Existing data warehouse tables can be identified to WhereScape RED. The tables are seen and can be managed to a degree. The main disadvantage is the increased difficulty in utilizing these tables when trying to extend the data warehouse. This option is however significantly quicker and easier than a rebuild. Refer to **Integrating** for details.

The decision as to which option to choose will depend on the size and complexity of the existing data warehouse. Another important factor is the degree to which the existing data warehouse is to be extended. If future enhancements revolve around new analysis areas that have little overlap with the existing environment then a integrate may be the best answer. If the data warehouse is small and relatively simple than a rebuild may be worth considering. In any event the best plan may be to do a test integrate and then re-evaluate the situation.

Rebuilding

The rebuild process essentially is a total re-creation of the data warehouse. One of the major impacts of such an approach is the **end user layer**, or rather the effect on the end user tools and saved queries and reports that are currently in use. The redesign or redeployment of this interface to the end users of the data warehouse may be too large a task to undertake. The problem can be circumvented to some degree though the use of views to make the new data warehouse environment look the same as the previous. But it is this impact and the subsequent re-testing process that must be considered when deciding to undertake a rebuild.

The advantages of a rebuild is the seamless integration of future analysis areas into the data warehouse and the single point of management that is provided. The major steps in the rebuild process will depend very much on the environment being replaced. As a guideline the following steps may be worth considering.

The rebuild process

1. Load a copy of the WhereScape metadata repository into an otherwise empty test environment. Refer to the RED Installation Guide for instructions on how to load a metadata repository.
2. Ensure there are no public synonyms that point to existing table names, if the rebuild process is to use the same names as some or all of the existing tables.

Working within the WhereScape RED tool proceed to:

1. Create connections to the new test data warehouse and to all the source systems.
2. Using the source system knowledge from the existing data warehouse, create the appropriate load tables in the data warehouse based on the existing extract or load methodology.
3. Build up the dimension, stage and fact tables using the same column and table names where possible.
4. Examine the existing procedures or update methodology and include this into the generated stored procedures.
5. Test the new environment.

© Copyright, WhereScape Software Ltd All rights reserved.

6. Work out a plan to convert the existing data into the new data warehouse. Where possible it is best to keep existing key values and re-assign sequences to match these existing key values where appropriate.
7. Convert and test the old data warehouse data in the new environment.
8. Redeploy the end user tool access.

Integrating

The integrate process

The steps in the integrate process are:

1. Create a test environment (database user) with the existing data warehouse tables loaded.
2. Load a copy of the WhereScape metadata repository into this test environment. Refer to the RED Installation Guide for instructions on how to load a metadata repository.

Working within the WhereScape RED tool proceed to:

1. Create any connections to UNIX or Windows servers where host scripts are currently executed. Refer to either **Windows connection** or **Unix connection** for details.
2. Create a Data Warehouse connection mapping back to the test environment. Refer to the **data warehouse / metadata repository connection** for details.
3. Incorporate any Host system scripts currently used. Refer to **integrating host scripts** for details.
4. Browse the Data Warehouse connection (**Browse > Source Tables**).
5. Drag and drop each existing data warehouse table into an appropriate object type. Refer to **integrating selecting table type** for details
6. Answer the retrofit questions, and build any required procedures. Refer to **integrating questions** for details.
7. Edit and amend all generated procedures, or create new procedures to handle the existing processing methodology. Refer to **integrating procedures** changes for details.
8. Test the new environment.

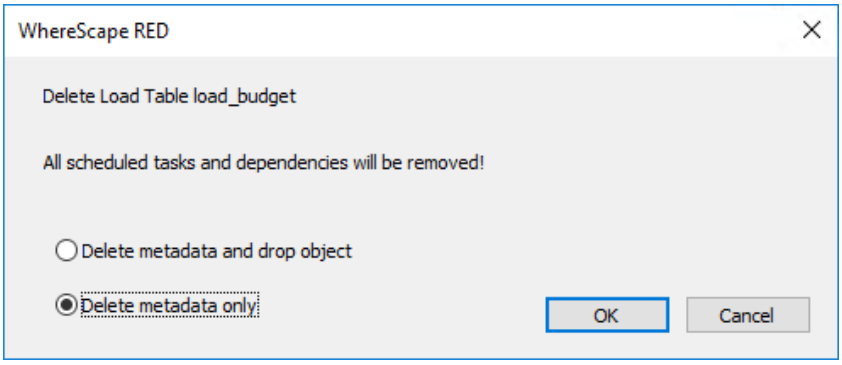
© Copyright, WhereScape Software Ltd All rights reserved.

Removing the metadata for a table

It is possible to delete the metadata for a table without deleting the table object itself. For example, if the integrate process is incorrectly undertaken, the metadata for the specific table can be removed.

To delete the metadata only:

1. Right click the table and select **Delete Metadata and Drop Table** from the context menu. A prompt is displayed which enables you to select the **Delete metadata only** option.



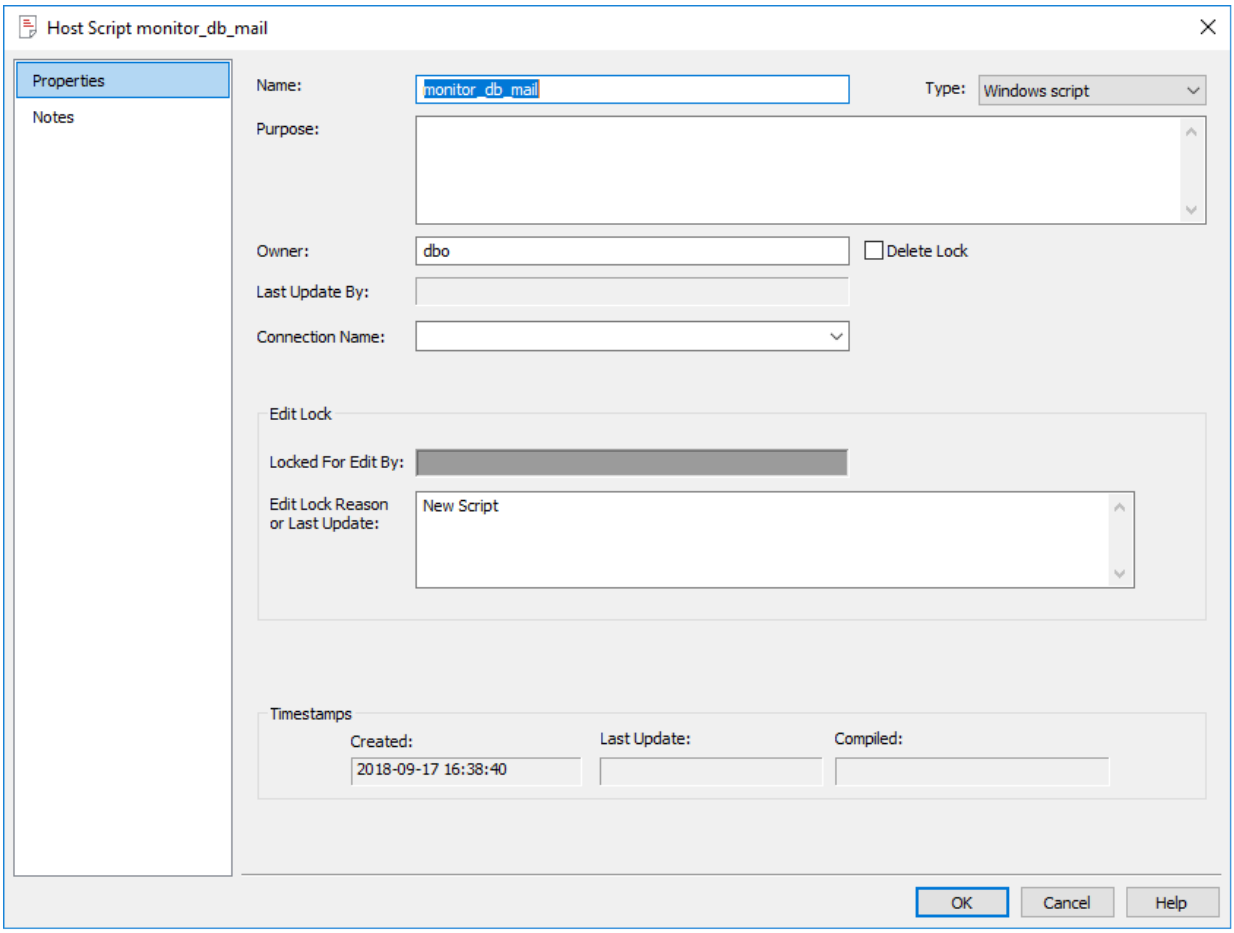
2. Ensure you have selected the **Delete metadata only** option and before clicking **OK**.

Integrating, Host Scripts

Existing host scripts (either UNIX or Windows) can be brought into the WhereScape RED metadata. To incorporate an existing script the process is as follows:

1. Create a **Host Script** object using WhereScape RED. In the left pane, right-click a project or the **All Objects** project and select **New Object**.
2. In the new object window, select **Host Script** from the Object Type drop-down and the type in a name for the new script.
3. Click **Add**, the following **Properties** window appears. Select the script type; either UNIX or Windows script. Select the appropriate connection from the **Connection Name** drop-down. Fill in the **Purpose** field to define the role of the script and then click **OK** to save the changes.

© Copyright, WhereScape Software Ltd All rights reserved.



4. Double-click the new script or right-click the new script and select **Edit the Script**.
5. Within the script editor, either paste the script or if it is available in your PC running RED, select the **File > Insert from File** option and load the file.
6. The script needs to be modified to handle the standards required by the RED scheduler. Refer to **Script based load** for details.

Integrating, Selecting a Table Type

When integrating existing tables there may not be a clear decision as to which table type to use. As a guideline, the following groupings can be considered.

Temporary tables:

- Load tables
- Stage tables

Permanent tables:

- Dimension tables
- Fact tables
- Aggregate tables

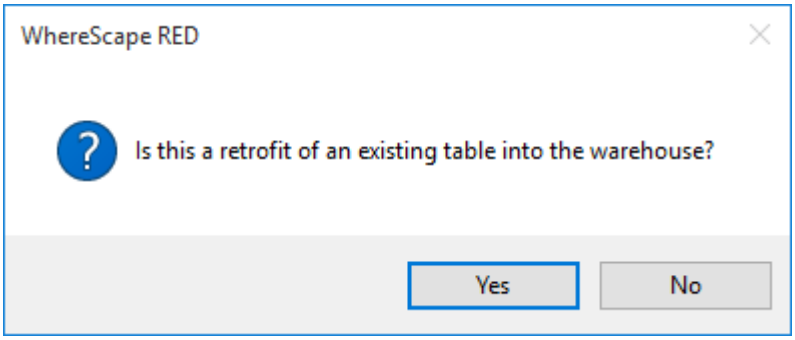
Although these table groups have very distinct names in terms of data warehousing, they do not impose any restrictions on the types of tables they contain. The table groupings are most relevant in the automatic generation of procedures, and in the sequencing for the scheduler.

Typically a mapping table may be stored in the Dimension table group.

© Copyright, WhereScape Software Ltd All rights reserved.

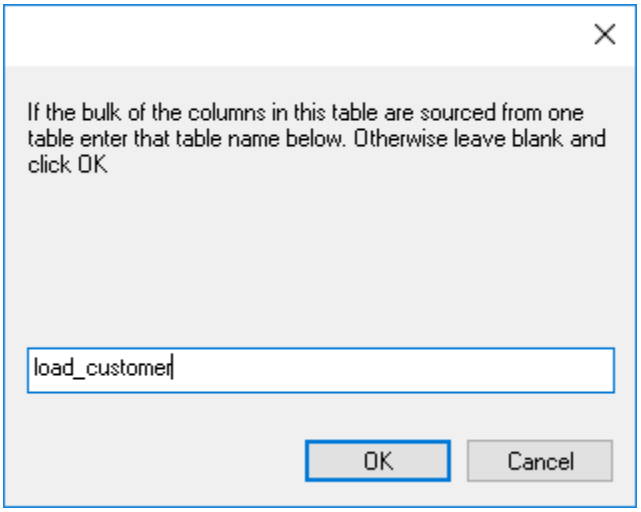
Integrating, Questions

When a table within the data warehouse schema, that is unknown to WhereScape RED, is dropped onto a table target the following dialog appears.



If this is an integrate then click **Yes** to proceed with the integrate process. The standard **New Object** dialog appears and it would be advisable to leave the name of the object unchanged so that it matches the existing table.

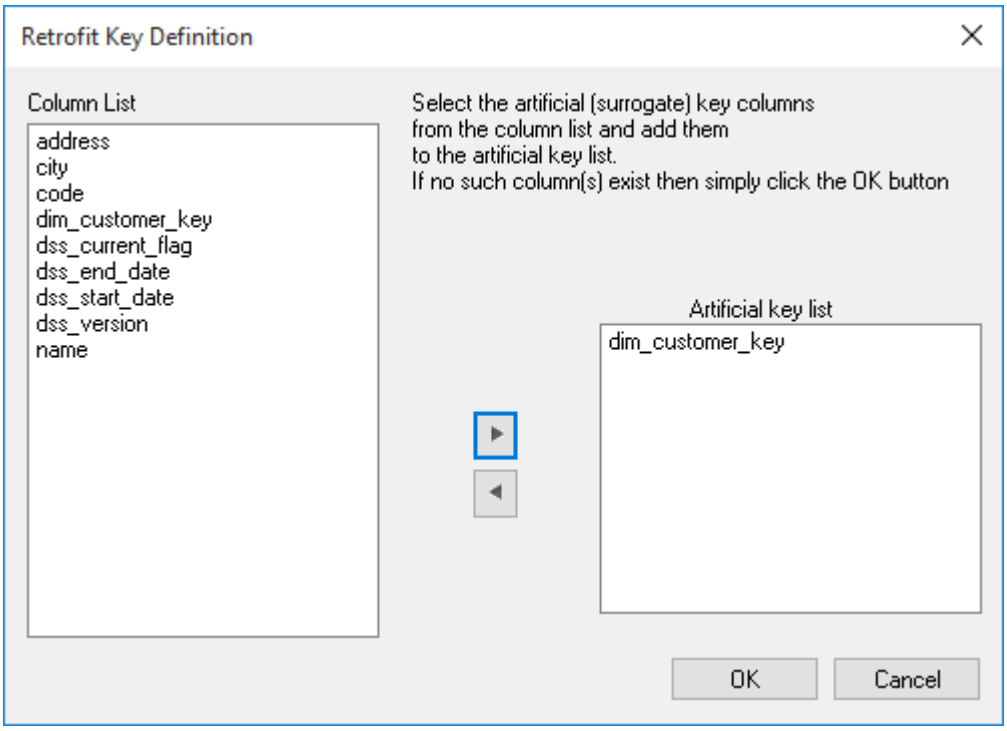
A dialog asks if the bulk of the columns in this table are derived from another table. If they are enter the table from which these columns derive at this stage. The purpose of this dialog is simply to set the **Source Table** field against each of the columns for the table.



© Copyright, WhereScape Software Ltd All rights reserved.

Artificial Key definition

If the table target is a dimension or fact table then the following dialog appears to enable the definition of any artificial or surrogate key. If no such key exists then simply proceed to the next question.



Get Key Function

If the target table is a dimension you will be asked if you want to generate a get key function. The get key function is used by WhereScape RED for rapid prototyping. It returns the artificial key when passed the business key. Although this function is possibly not necessary in the existing environment it is useful if the dimension is to be used in any expansion of the data warehouse. If an existing get key type function exists, then this can be used, although changes will probably need to be made to the generated procedures to utilize this existing get key function.

Note

The **Get Key** function is not available for dimension objects that are stored in custom database targets.

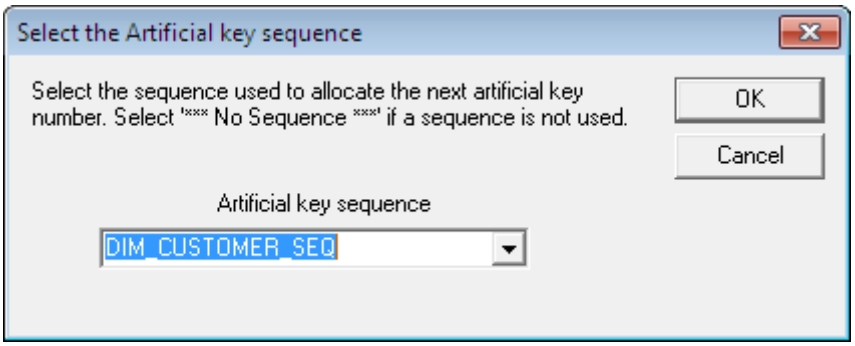
Update Procedure

If the target table is a dimension, stage, fact or aggregate a dialog box will ask if an update procedure or update template is required. If selected a subsequent dialog will define the structure and content of this procedure. If an existing procedure is to be modified to include the scheduler imposed procedure standards it is best to select this option and use the existing procedure name to cut down on the amount of work required. The existing procedure can then be loaded into the generated procedure using the procedure editor.

© Copyright, WhereScape Software Ltd All rights reserved.

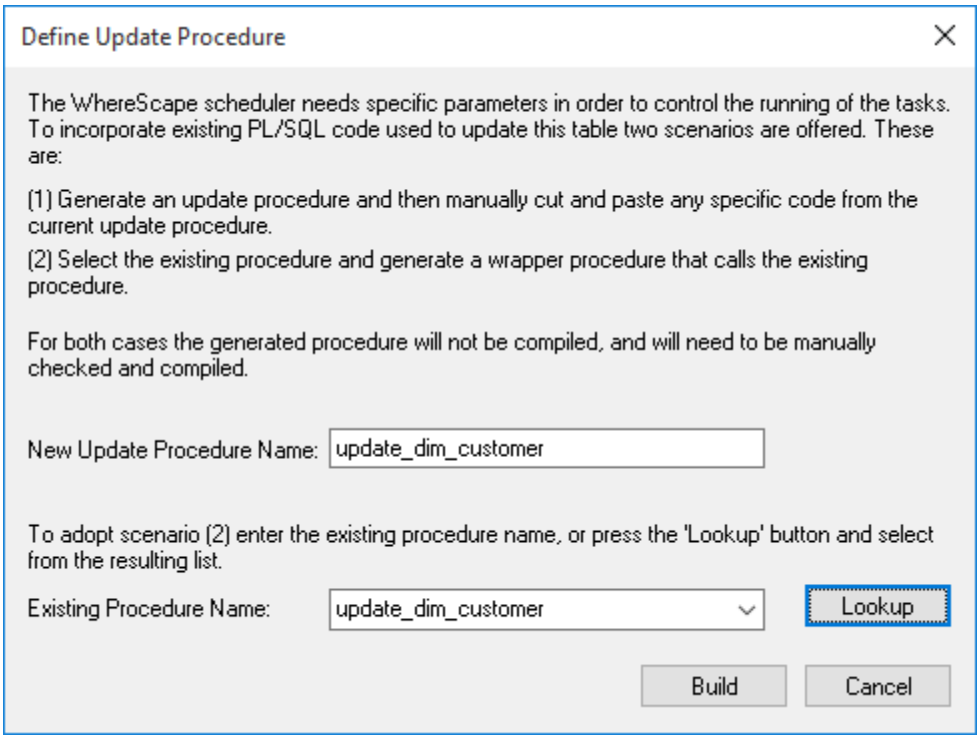
Artificial Key Sequence

If a get key function or update procedure is to be built and an artificial key is in use for the table then a dialog will prompt for the artificial key sequence. A drop-down list provides all the sequences defined for this database user. This sequence is an Oracle sequence that is used in the automatic generation of the artificial key. If no such sequence is used then select the *** no sequence *** option.



Update Procedure Definition

If an update procedure was selected then the following dialog will display to help define how the update procedure is generated.



As defined in this dialog box the scheduler needs specific parameters in order to control and run a procedure. An existing procedure will not have these parameters. There are basically two forms of generated procedure. The first option is to generate a procedure that looks much like a standard WhereScape RED procedure and then manually insert the changes from the existing procedure. The second option is to generate a wrapper procedure that calls the existing procedure. This second option will still require manual changes to the procedure to return a valid return code and message.

Business Key definition

If a get key or update procedure has been selected a dialog will now prompt for the selection of the business key from the table. Multiple columns may constitute the business key, but they must uniquely identify each record in the table.

Index definition

Any indexes associated with the table will now be automatically defined and loaded into the metadata. Changes may need to be made in terms of when the indexes are dropped and to set the **Drop Before Update** checkbox if appropriate and the scheduler is to be used to manage these indexes.

© Copyright, WhereScape Software Ltd All rights reserved.

Procedure creation

If defined the get key and update procedures will now be generated. They will need to be manually edited and compiled. Refer to **Integrating procedures** for details.

Integrating, Procedures

The procedures managed by the WhereScape scheduler require the following standards.

Parameters

The procedure must have the following parameters in the following order:

Parameter name	Input/Output	Oracle Type	SQL Server/DB2 Type
p_sequence	Input	Number	Integer
p_job_name	Input	Varchar2	Varchar(256)
p_task_name	Input	Varchar2	Varchar(256)
p_job_id	Input	Number	Integer
p_task_id	Input	Number	Integer
p_return_msg	Output	Varchar2	Varchar(256)
p_status	Output	Number	Integer

The input parameters are passed to the procedure by the scheduler. If the procedure is called outside the scheduler then the normal practice is to pass zero (0) in the sequence, job_id and task_id. A description of the run can be passed in the job name and the task name is typically the name of the procedure.

The output parameters must be populated by the procedure on completion. The return_msg can be any string up to 256 characters long that describes the result of the procedures execution. The status must be one of the following values:

Status	Meaning	Description
1	Success	Normal completion
-1	Warning	Completion with warnings
-2	Error	Hold subsequent tasks dependent on this task
-3	Fatal Error	Hold all subsequent tasks

The major task in integrating a procedure will be in adapting it to the WhereScape scheduler standards and work flow.

Integrating, Views

When integrating views an additional step is required if you want WhereScape RED to be able to recreate the view.

The view will be mapped correctly and the **Get Key** function can still be built. This step is only required if the view is to be re-created.

Note

The **Get Key** function is not available for dimension objects that are stored in custom database targets.

Change the source column on the artificial key to match the artificial key in the table from which the view was created.

© Copyright, WhereScape Software Ltd All rights reserved.

Integrating, WhereScape Tables

When integrating WhereScape generated tables and views a number of additional considerations need to be taken.

The Dimension tables and views make use of secondary business key columns such as dss_source_system_key, dss_current_flag, dss_end_date, dss_version to assist in handling various forms of slowly changing dimensions and non conformance. These secondary business keys need to be flagged as such against the dimension.

Change the properties of all such columns. The key type should be set to 1, and the primary business key checkbox should be set.

© Copyright, WhereScape Software Ltd All rights reserved.

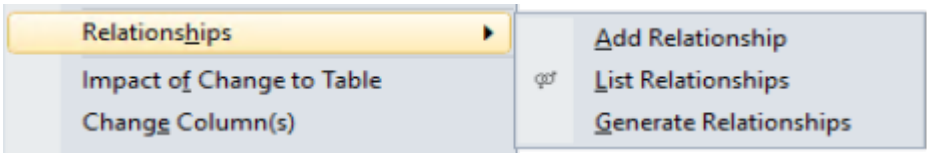
Relationship Maintenance

Relationship Maintenance is available for the maintenance of joins between objects; providing a way to record joins between tables when surrogate keys are not being used. This functionality then enables the generation of **Links Diagrams** for these objects.

Note

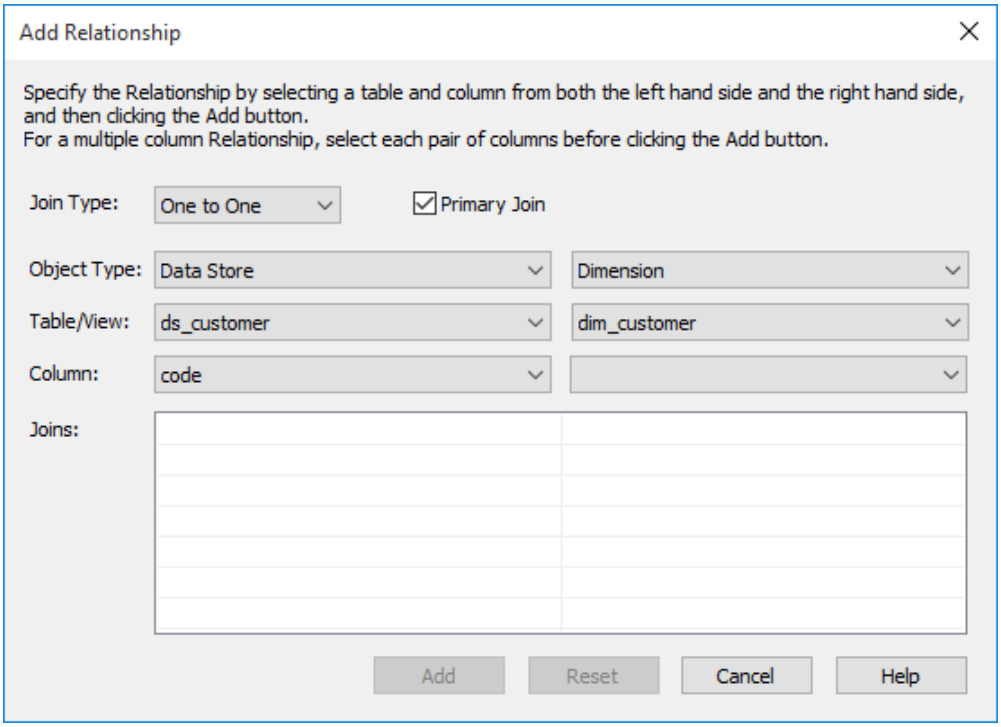
It is necessary to explicitly specify relationships for tables on **Tabular** target databases for the relationships to be created in the Tabular database.

Relationship Maintenance options are available in the Relationships sub-menu when right-clicking on an object in the **Object Pane**.



Add Relationship

To add a relationship, right-click on the object in the **Object Pane** and select **Relationships->Add Relationships**. The following dialog is displayed.



For each object in the relationship, enter in the following details:

Options	Description
Join Type	For the **Join Type**, choose between the following: • Undefined • Many to One • One to One • One to Many • Many to Many

© Copyright, WhereScape Software Ltd All rights reserved.

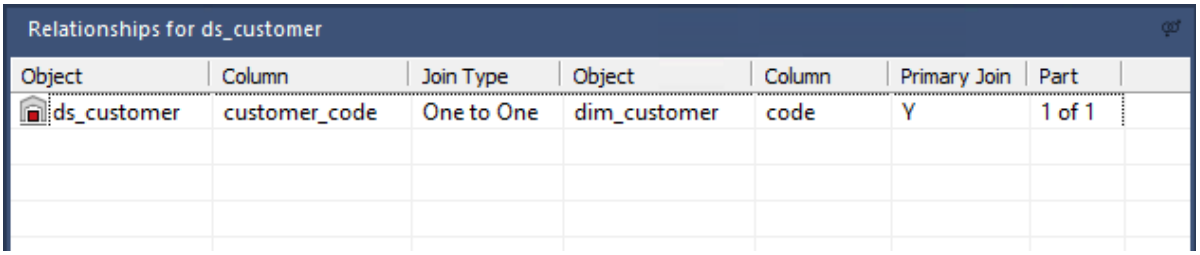
Options	Description
Primary Join	If this is a primary join, select the **Primary Join** check-box. For MSAS Tabular tables this defines an active relationship. At most, one relationship between two specific tables can have this enabled.
Object Type	Enter the **Object Type** from the drop-down box (Data Store, Load table, Stage table, etc.).
Table/View	Enter the name of the object in the relationship.
Column	Enter the **Column** to join in each object.

Once you have entered the details for the join, the joined columns are displayed in the list of **Joins** at the bottom of the dialog box. Erroneous joins can be removed by right-clicking on the join and selecting **Remove Join**. All joins can be removed by clicking the **Reset** button.

To add all the relationships shown in the list of **Joins**, click the **Add** button.

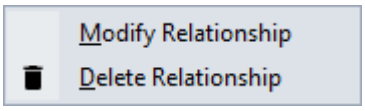
List Relationships

To view relationship for an object, right-click on the object in the **Object Pane** and select **Relationships > List Relationships**. The relationships for the selected object are displayed in the **Drop Target Pane** (middle pane).



Multi-column joins are shaded when one join is selected.

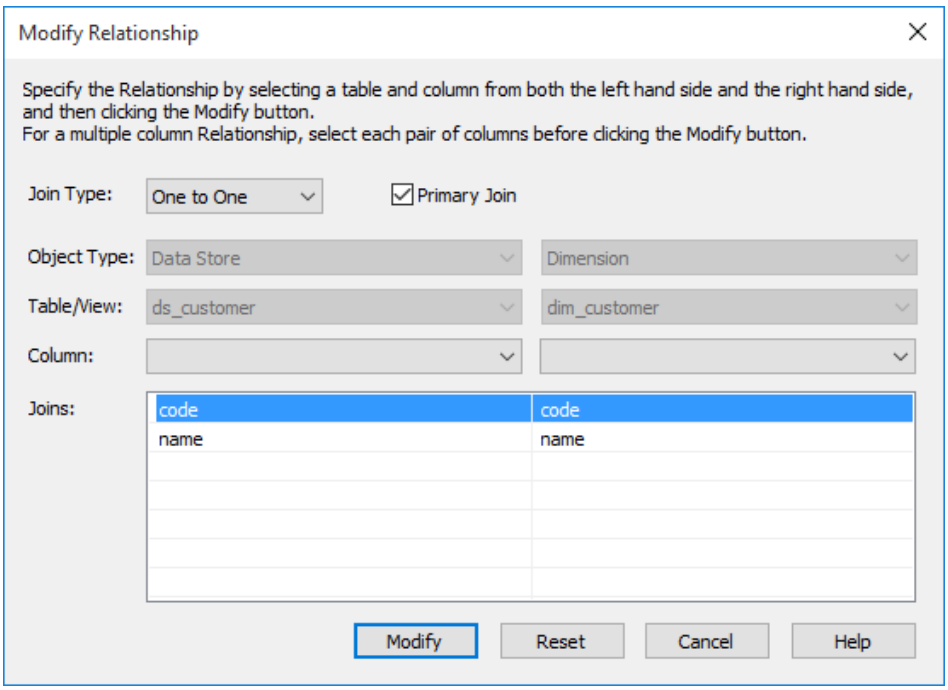
Right-clicking on a column or join displays the following menu:



© Copyright, WhereScape Software Ltd All rights reserved.

Modify Relationship

The Modify Relationship option shows the following dialog, allowing the editing of joins (including multi-column joins) between the two objects in the selected relationship.



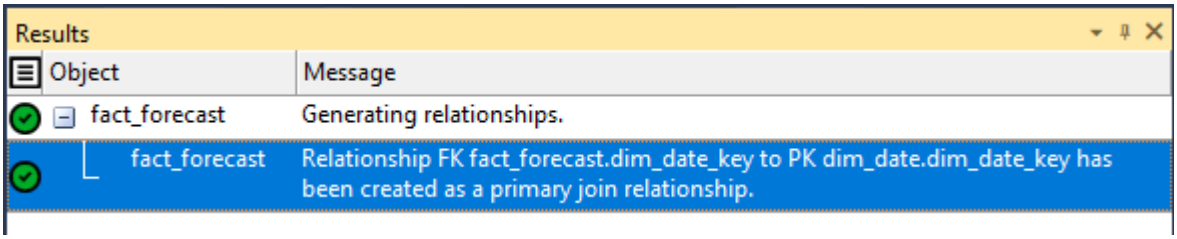
Relationships are edited in this dialog in the same way as the **Add Relationship** dialog above. **Object Types** and **Table names** cannot be modified.

Delete Relationship

Deletes the selected relationship.

Generate Relationships

To generate relationships in metadata for an object, right-click on the object in the **Object Pane** and select **Relationships > Generate Relationships**. Results are shown in the **Results Pane**.



© Copyright, WhereScape Software Ltd All rights reserved.

Upgrading RED

WhereScape Versions

WhereScape RED has a four part version number normally shown as xx.xx.xx.xx. An example of this may be 6.0.6.0. The first number represents the major release. The second number represents the metadata repository version. The third and fourth numbers relate to application specific releases.
From the example above we see that the current version is version 6 of WhereScape RED. We are on version 6.0 of the metadata repository.

Metadata Changes

A change from a 6.0.. release to a 6.5.. release would indicate a change in the metadata tables. All metadata changes are non destructive. They simply add new columns or new meta tables. In this way they can be applied without harming an existing metadata repository. The impact of a metadata change is that the associated applications (namely the RED executable, the Scheduler, security module etc.) will need to be at the same metadata version. Therefore a 6.0.6.0 version of RED may not successfully run against a version 6.5 metadata repository.

Application changes

The final two numbers in the version represent application releases. Applications are deemed to be all of the executable images supplied with WhereScape RED as well as the UNIX scheduler scripts and the stored procedures. Application changes reflect enhancements and bug fixes. A change in the first number indicates a major enhancement in one of the application components.

Upgrading RED

Upgrading WhereScape RED consists of the following steps:

1. Allow any active jobs to complete. Halting active jobs will allow running tasks to complete with no new tasks starting. Aborting active jobs will kill any running tasks and stop running jobs.
2. Stop all schedulers. Windows schedulers can be stopped with WhereScape's Setup Administrator. To stop a UNIX or Linux scheduler, kill the active scheduler process and comment their crontab entries (to stop the scheduler re-starting itself).
3. Close any WhereScape programs that are running on your machine.
4. Install the new version of RED on your machine.
5. Back up your metadata.
6. In RED Setup Administrator, select the Validate Metadata Repository option. This function may re-compile existing or create new metadata procedures; metadata tables may be altered or new tables created.
   When performing an upgrade of RED:
   - Click **OK** when prompted to validate metadata tables.
   - Click **Yes** when prompted to re-create metadata views.
   - Click **Yes** when prompted to re-compile metadata procedures.

   Warning

   If the above steps are not all completed during an upgrade the metadata may be left in an inconsistent state.

7. For **Oracle** metadata updates see the note below on **recompiling Oracle invalid procedures** during a RED upgrade.
8. Back up your metadata again (just in case).
9. If using a UNIX or Linux scheduler and this is a major application enhancement then rename the wsl/bin directory to say wsl/bin_versionxxxxxx. Create a new bin directory and ftp over the files under the unix directory (see the main install instructions). Change the protections on the files (chmod 750 *.sh).

© Copyright, WhereScape Software Ltd All rights reserved.

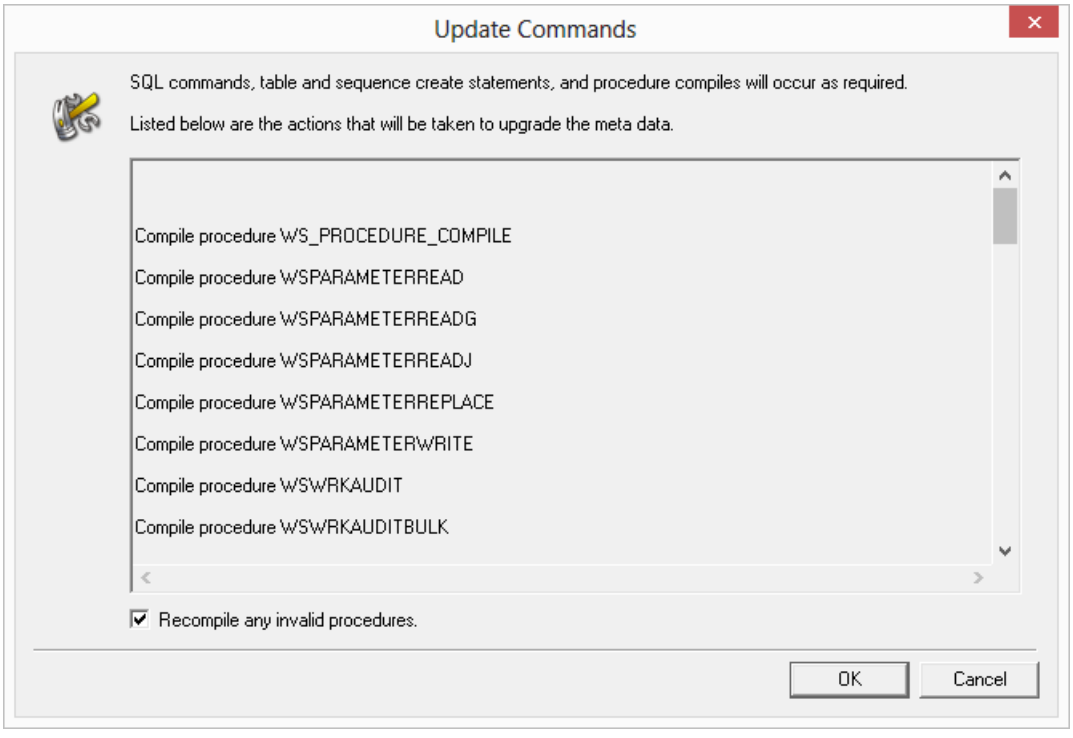
10. Restart all schedulers. Windows schedulers can be restarted with WhereScape's Setup Administrator. For any UNIX or Linux schedulers, uncomment any commented out crontab entries - this is enough to restart the schedulers.
11. Restart any halted or aborted jobs.

Notes

1. There are different versions of the scripts for each database and for UNIX versus Linux. You should also be looking in the sub directory with the highest version number.
   Oracle When using a UNIX/Linux scheduler for Oracle, It is recommended that users copy the new meta_backup_680.sh file to use Datapump's expdb/impd instead the deprecated exp/imp tools. This version uses the data pump export executables expdp/impdp. It assumes that the scheduler and the Oracle database reside on the same server.
2. Metadata tables do not change between minor releases, but metadata procedure often and usually do change.
3. RED will not let you sign into an old repository version, using a newer version of RED.
4. RED will let you sign into a new repository version, using an older version of RED, but it will warn you that this may potentially cause issues.
5. It is very important when using the Windows scheduler to have the installed RED version on the scheduler server EXACTLY matching the stored procedures.
6. Side by side installations are possible (two versions in two directories on the same machine), but be careful with schedulers. If you install the new version of RED in a new directory, you will have to remove and reinstall all windows schedulers in order for the scheduler(s) to use the new version of RED.
7. As a general rule, the UNIX and Linux scheduler scripts, metadata tables/procedures and RED front end should all be in sync.
8. If the scripts for the UNIX or Linux scheduler have changed, you should replace files in your old and new shell scripts in the WhereScape Program Files directory and see if any have changed.

Oracle invalid procedures:

When validating an Oracle Metadata Repository, tick the **Recompile any invalid procedures** check-box to have any invalid user procedures recompiled during a metadata validate.



© Copyright, WhereScape Software Ltd All rights reserved.

Login Checks

The following checks are performed during login; and if necessary, warning messages are displayed, see more about each specific warning by following the links below:

Login Data Source does not match the data warehouse connection's ODBC DSN

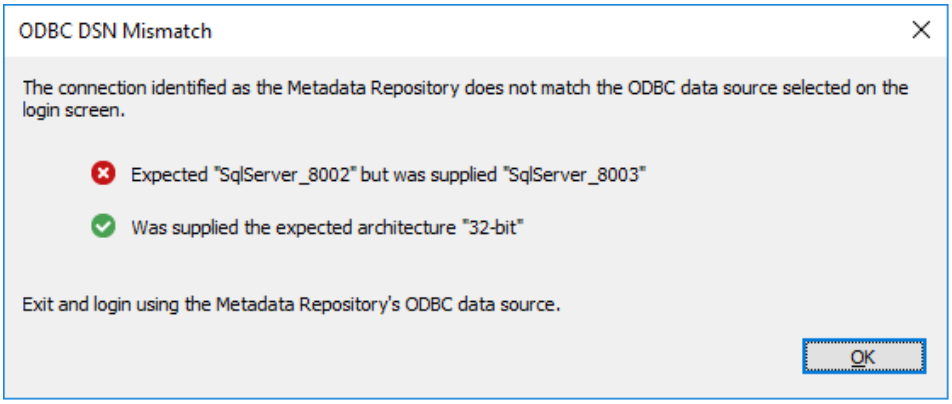
More than one connection has the data warehouse option enabled

Oracle Individual User

© Copyright, WhereScape Software Ltd All rights reserved.

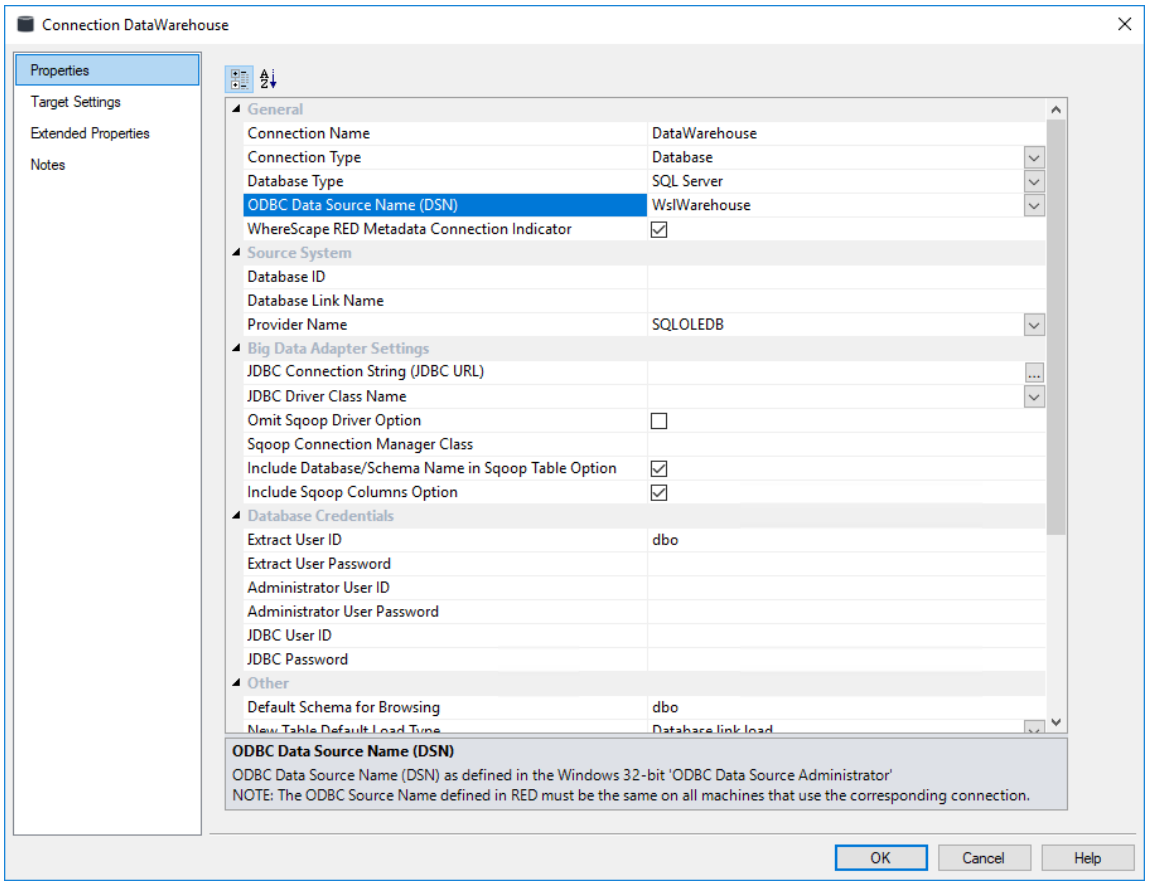
Data Source does not match the data warehouse connection's ODBC DSN

Warning about the login Data Source not matching the data warehouse connection's **ODBC DSN:**



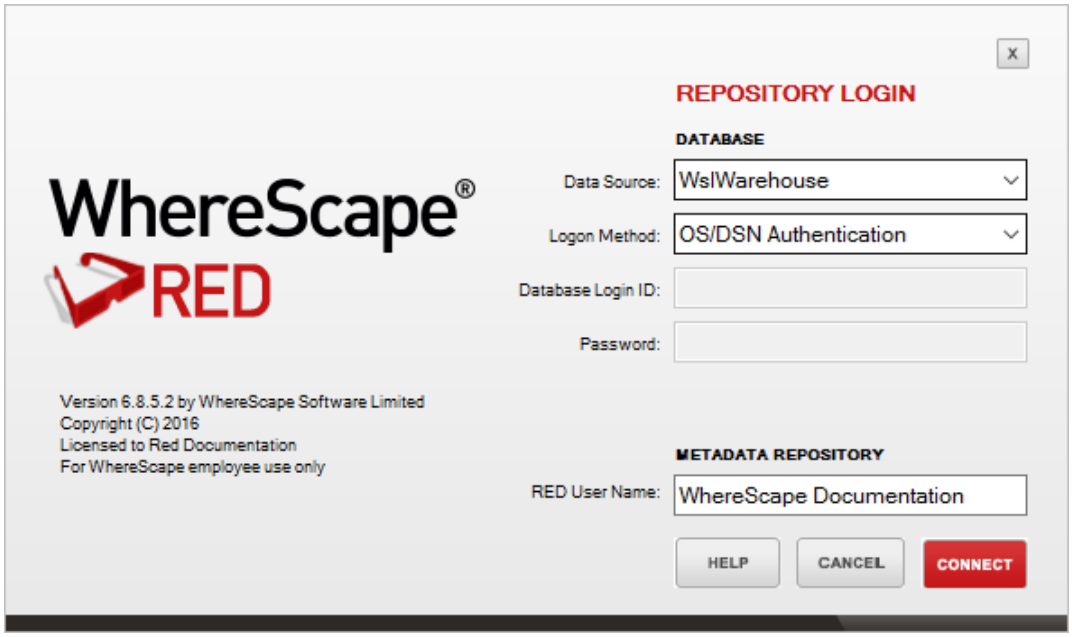
You can correct this issue by performing one of the following actions:

1. Alter the Data Warehouse Connection **ODBC Data Source Name (DSN)** to match the login **Data Source**.



© Copyright, WhereScape Software Ltd All rights reserved.

2. Logoff and log back in using the **Data Source** with the same name as the Data Warehouse Connection **ODBC Source**.

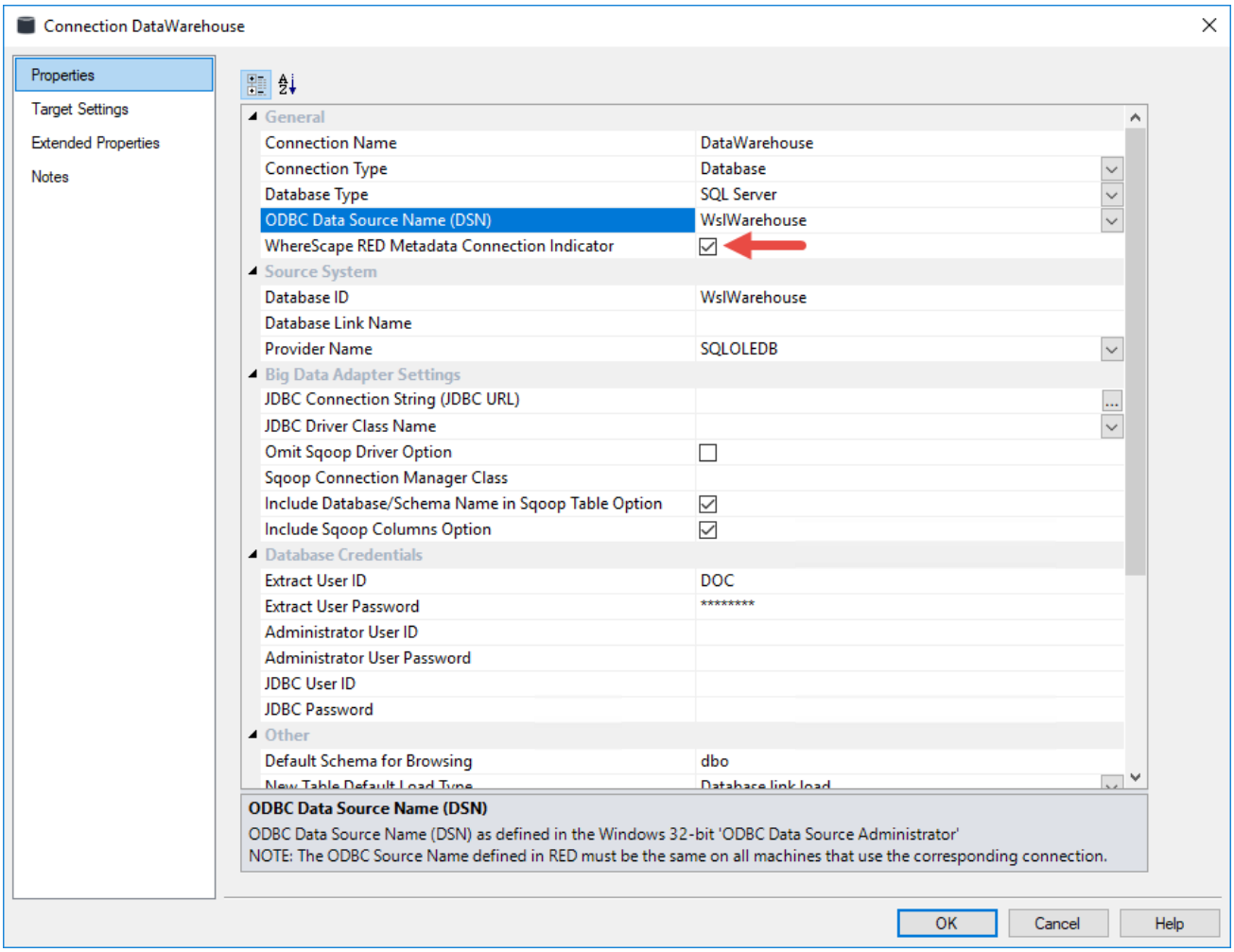


© Copyright, WhereScape Software Ltd All rights reserved.

More than one connection has the Data Warehouse Indicator Enabled

Warning about more than one connection having the data warehouse option enabled.

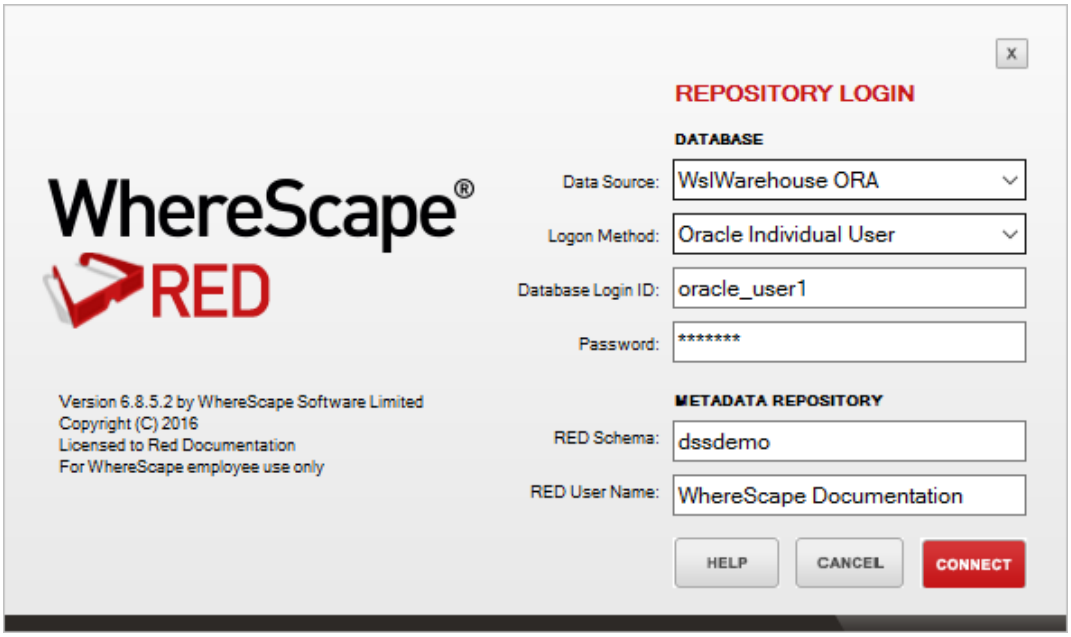
You can correct this issue by editing all the connections and making sure that only one is set to Data Warehouse:



© Copyright, WhereScape Software Ltd All rights reserved.

Oracle Individual User

For any errors or for more information about logging in, or creating an **Oracle Individual User**, refer to Creating an Oracle Individual user for details in the WhereScape RED Installation Guide.



© Copyright, WhereScape Software Ltd All rights reserved.

Data Type Mappings

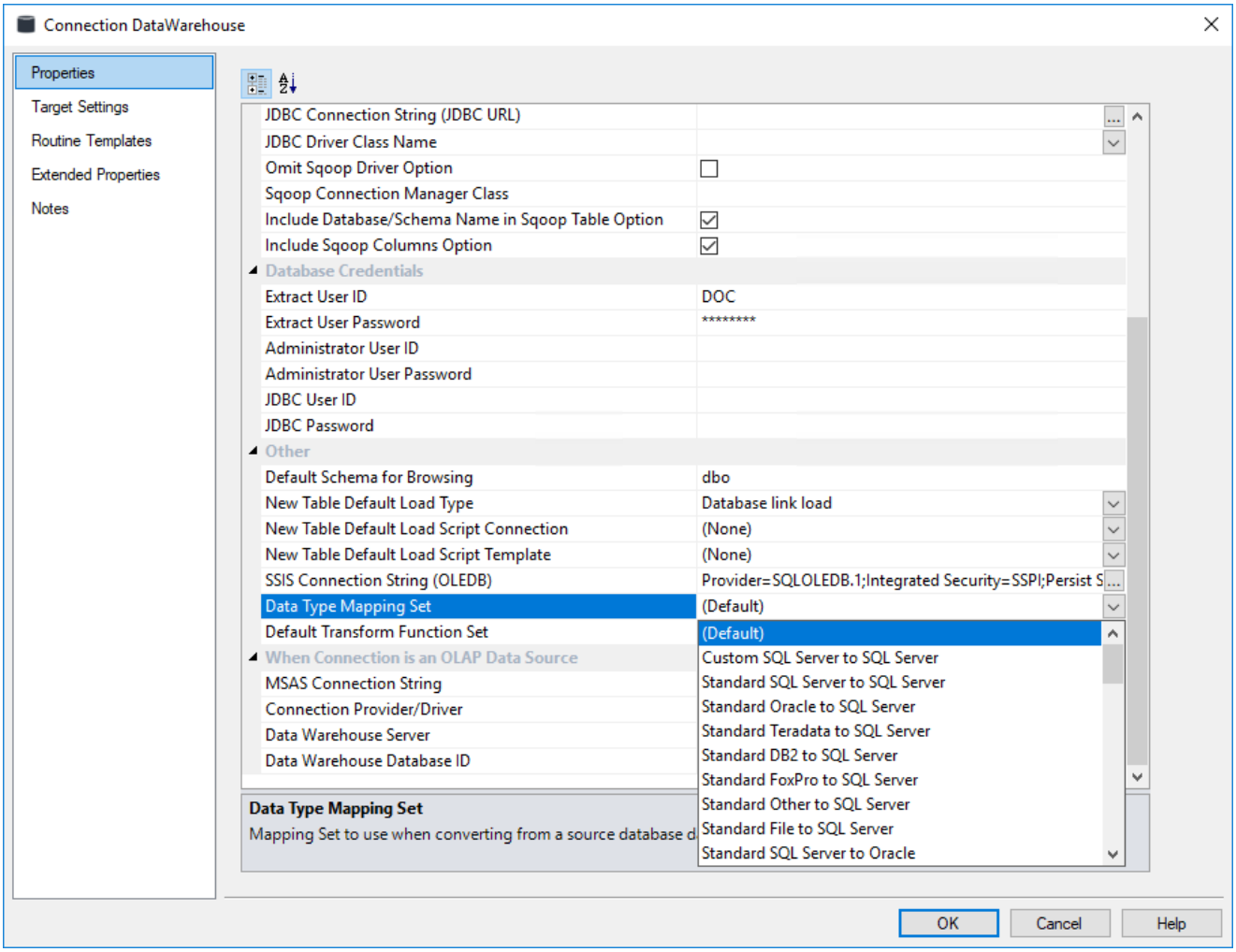
Using Data Type Mapping Sets

Data type mapping sets contain a list of mappings that are used when loading tables into the data warehouse.

Custom data type mapping sets give you the ability to automatically change the data type of any column or to add column transformations when dragging and dropping new Load tables. These mapping sets can be created, edited, deleted, imported and exported using the **Data Type Mappings** options from the **Tools** menu.

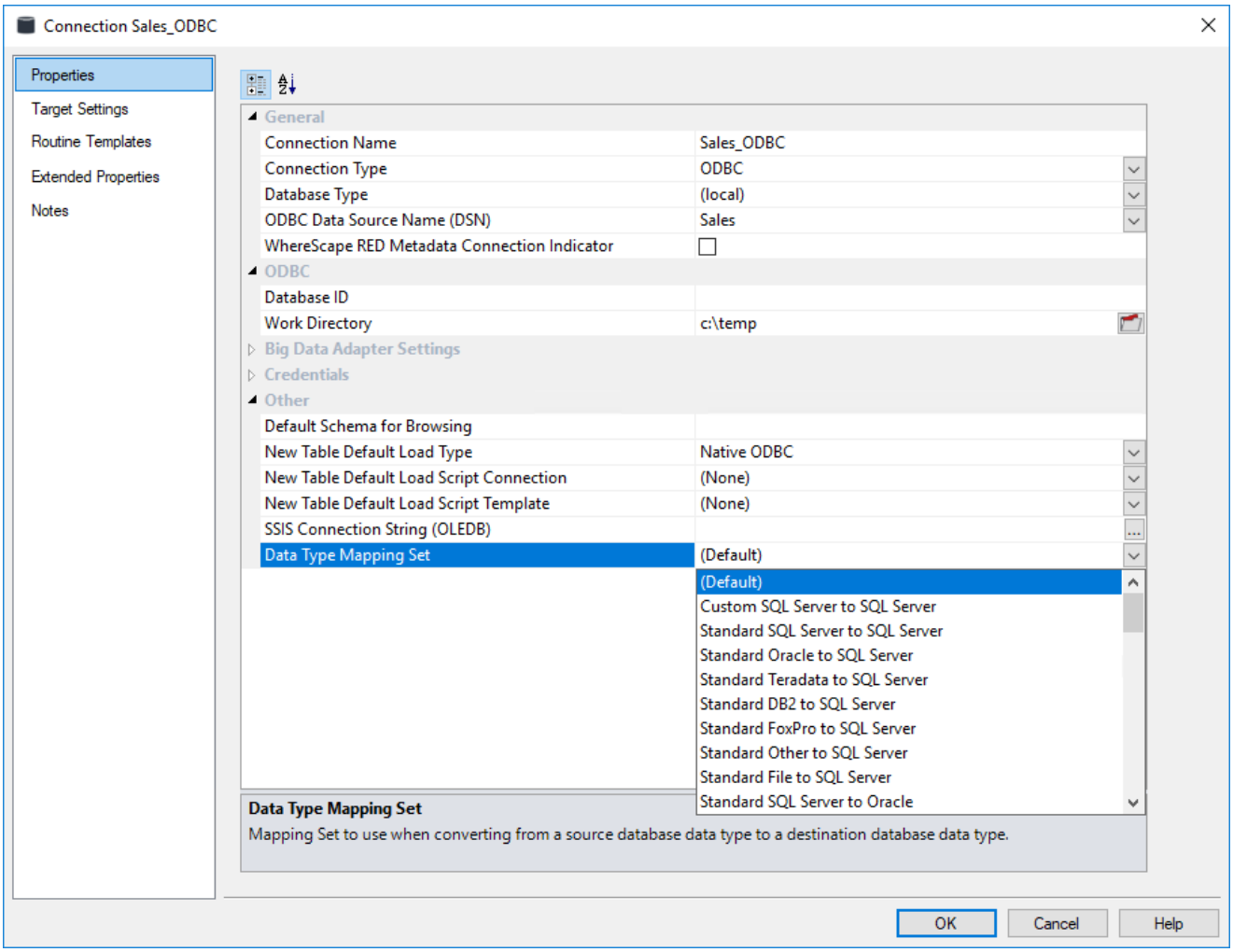
In the connection **Properties** screens, a drop-down list **Data Type Mapping Set** is displayed. This enables you to specify the default data type mapping set to use when loading tables into the data warehouse.

Data Warehouse Connection Properties



© Copyright, WhereScape Software Ltd All rights reserved.

Non-Data Warehouse Connection Properties

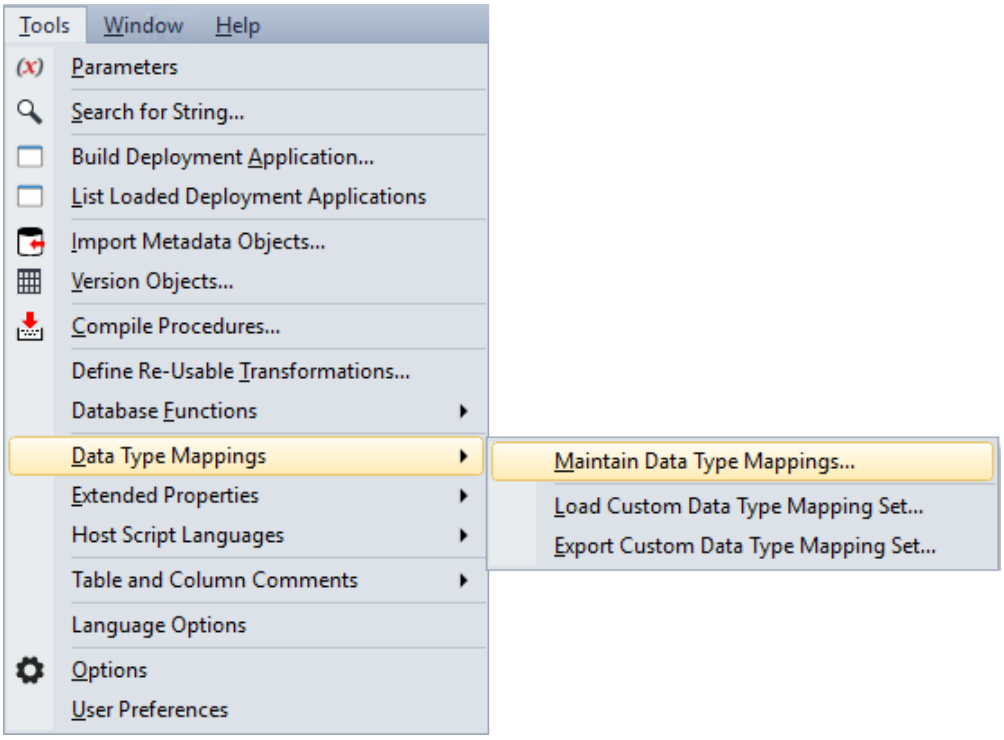


The succeeding sections describe how to manage data type mappings in RED.

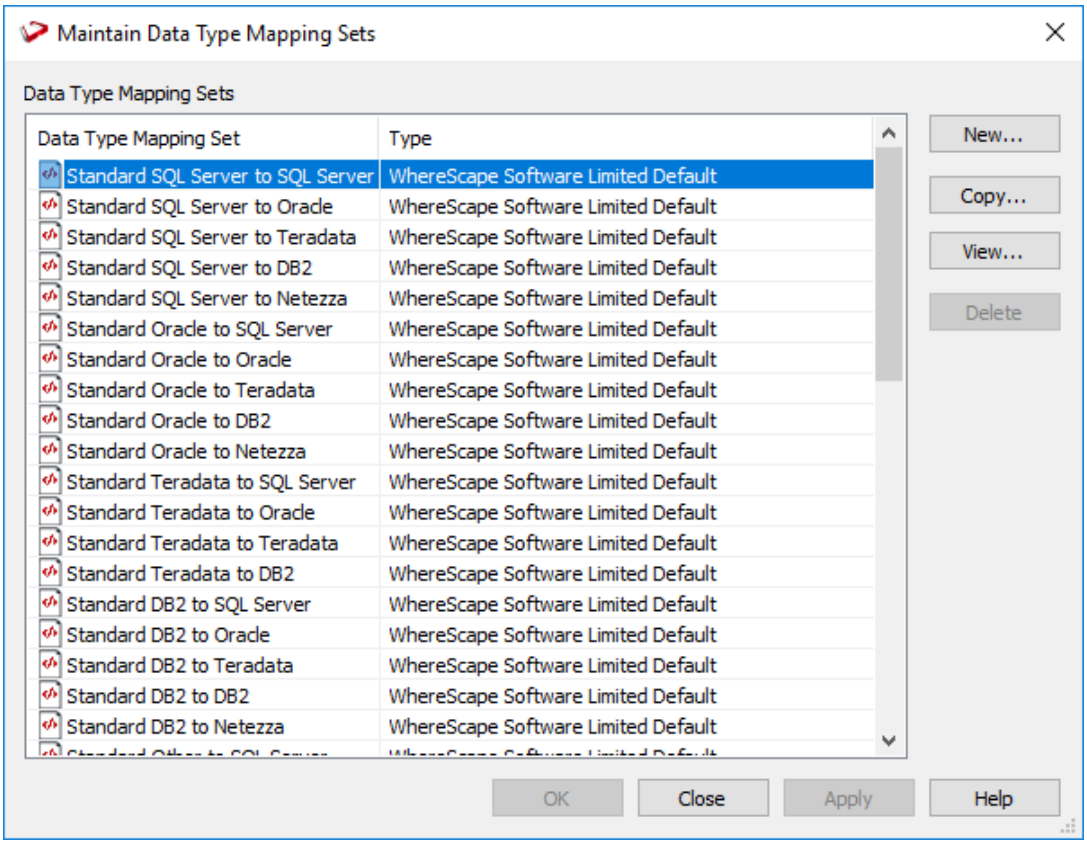
© Copyright, WhereScape Software Ltd All rights reserved.

Maintaining Data Type Mapping Sets

To maintain the data type mapping sets, select **Tools > Data Type Mappings > Maintain Data Type Mappings**.



The **Maintain Data Type Mapping Sets** window is displayed.



Select a data type mapping set from the **Data Type Mapping Sets** list.

In the list of standard files, only the files relevant to the database that you are working on is displayed.

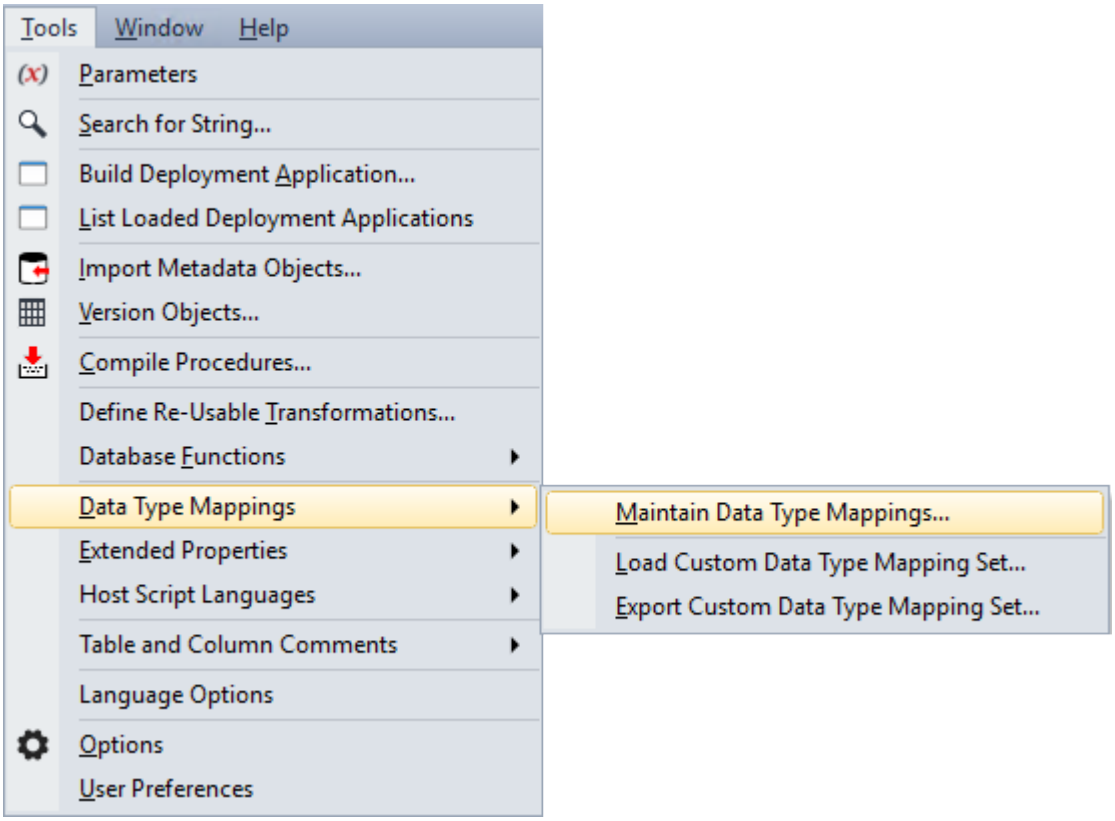
To create a data type mapping set, refer to **Creating a New Data Type Mapping Set**

© Copyright, WhereScape Software Ltd All rights reserved.

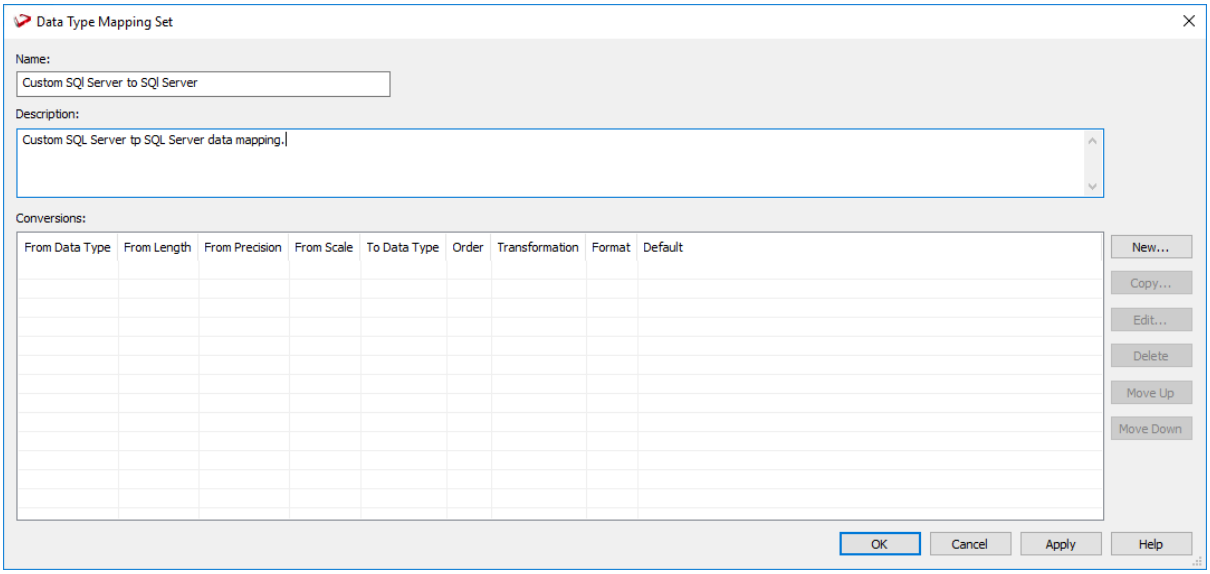
To copy a data type mapping set, refer to **Copying a Data Type Mapping Set**
To edit a data type mapping set, refer to **Editing a Data Type Mapping Set**
To delete a data type mapping set, refer to **Deleting a Data Type Mapping Set**

Creating a New Data Type Mapping Set

To create a new data type mapping set, select **Tools > Data Type Mappings > Maintain Data Type Mappings**.

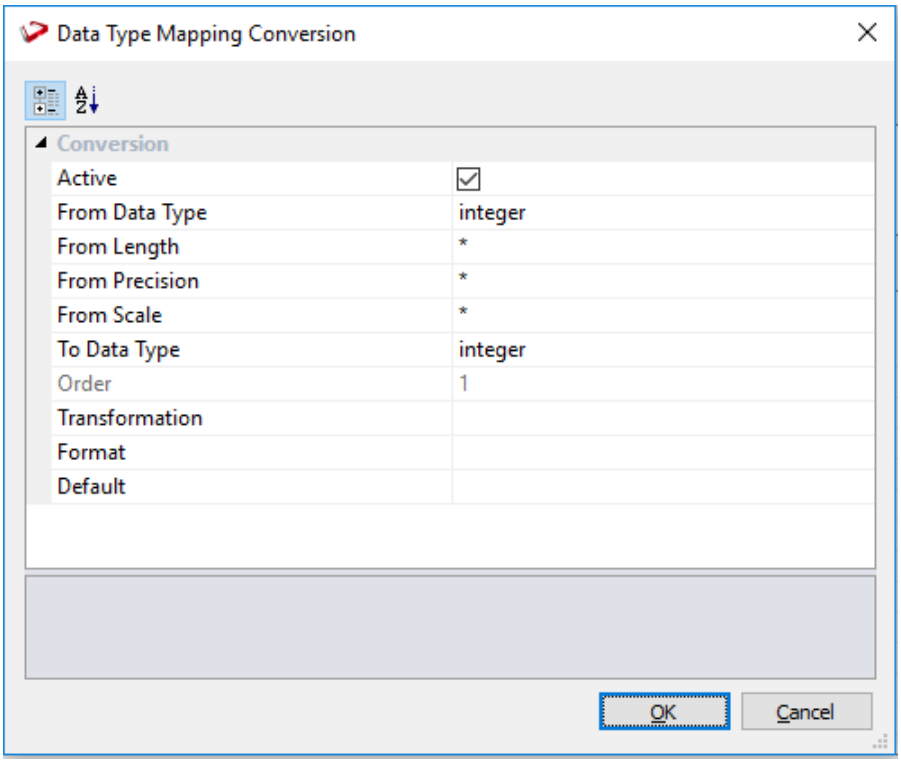


1. Click the **New** button from the **Maintain Data Type Mapping Sets** window to enter the name and description of the new Mapping Set.



2. You can then enter the individual data type mapping conversion parameters by clicking the **New** button on the right side of the window.

© Copyright, WhereScape Software Ltd All rights reserved.

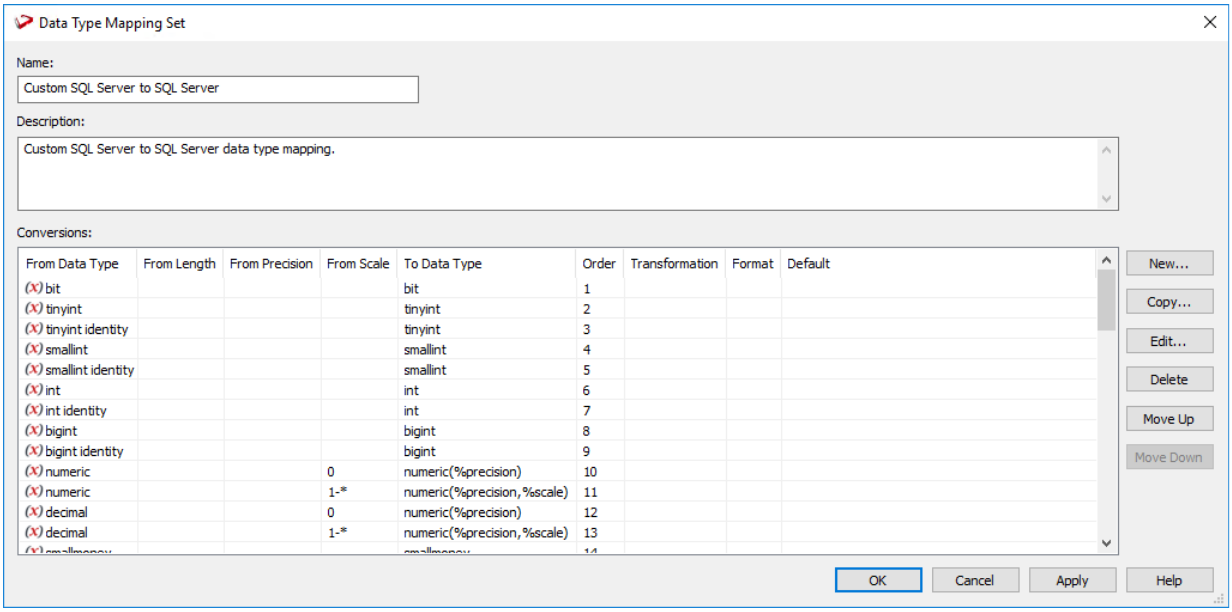


The following describe the conversion parameters:

Parameter	Description
Active	Indicates if the conversion parameter can be used when searching for a match.
From Data Type	The source Data Type to that is used for the conversion to the Target Data Type. It can be any type that source RDBMS uses.
From Length	The length parameter to match. (the number of characters or the number of bytes used to store the number).
From Precision	The precision parameter to match. (the number of digits in a number).
From Scale	The scale parameter to match. (the number of digits to the right of the decimal point in a number).
To Data Type	The data type, including parameters, which is the output of this mapping. Available tokens are %data_type, %length, %precision and %scale. Refer to **Custom Data Type Mapping Set Examples** for details.
Order	The order of the parameter within the enclosing set of the conversion.
Transformation	The transformation to apply when the conversion is applied. See **Custom Data Type Mapping Set Examples** for details.
Format	The format that values using the data type will use. Typically used for numeric values.
Default	The default value to use for values that use the data type.

© Copyright, WhereScape Software Ltd All rights reserved.

3. Click **OK** to save the parameter. Repeat to define more conversion parameters.

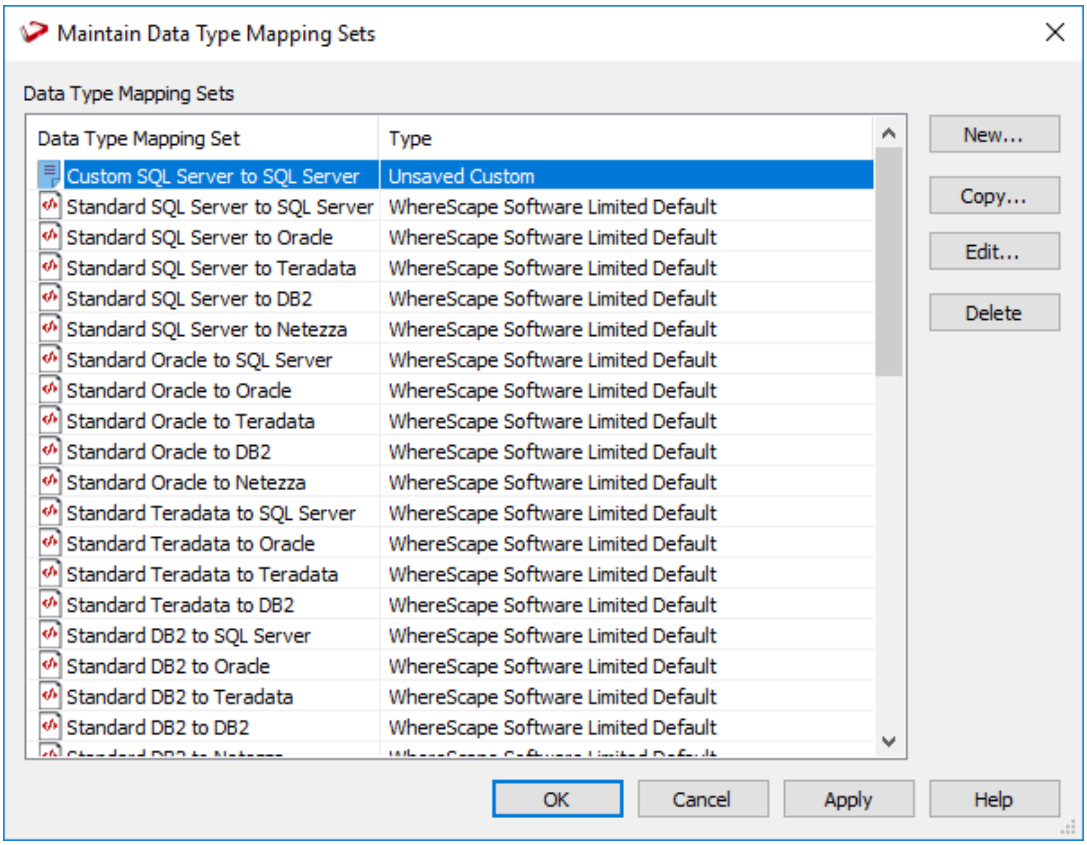


Notes

Use the buttons on the right side of the **Data Type Mapping Set** window to edit, copy or delete a selected parameter. To move a parameter up or down in the list, select it and then click the **Move Up** or **Move Down** button.

The order of the data type mapping conversion parameters is significant because when loading a table, the procedure checks the data type mappings from top to bottom and stops when a data type and its parameters are correctly matched. A blank parameter means that it will match to anything.

4. Click **OK** when you're done. The new data type mapping set is listed in the **Maintain Data Mapping Sets** window.

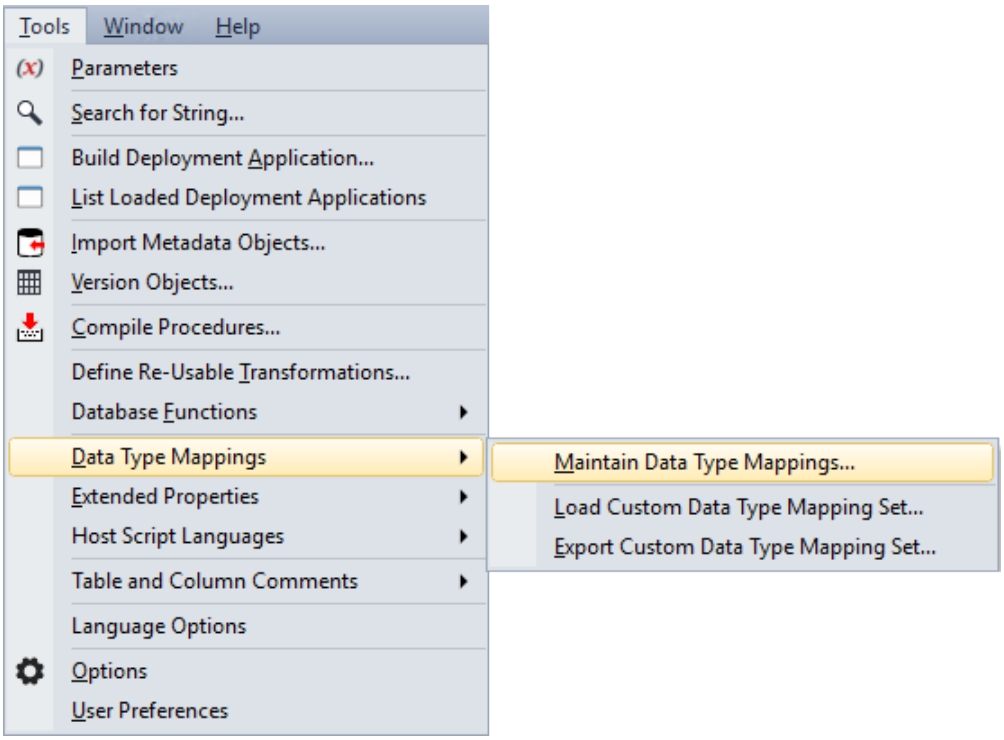


5. Click **OK** or **Apply** to save the new data type mapping set.

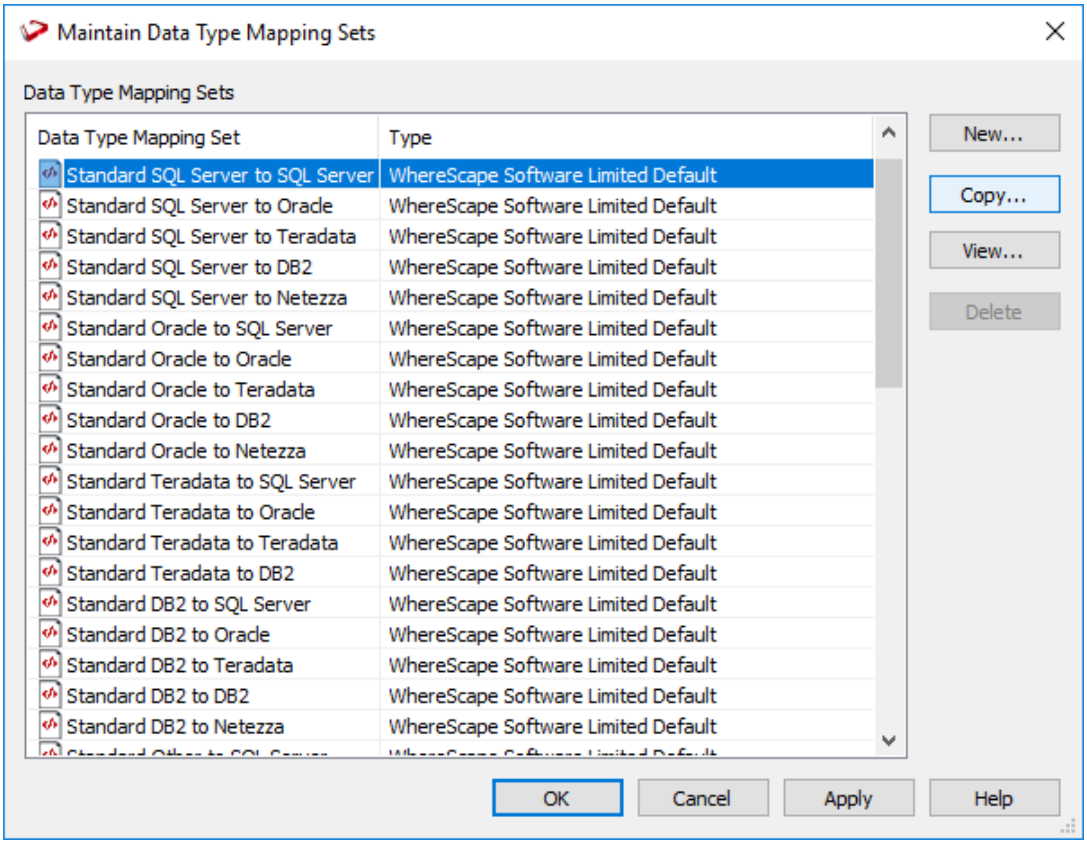
© Copyright, WhereScape Software Ltd All rights reserved.

Copying a Data Type Mapping Set

To copy an existing data type mapping set, select **Tools > Data Type Mappings > Maintain Data Type Mappings**.

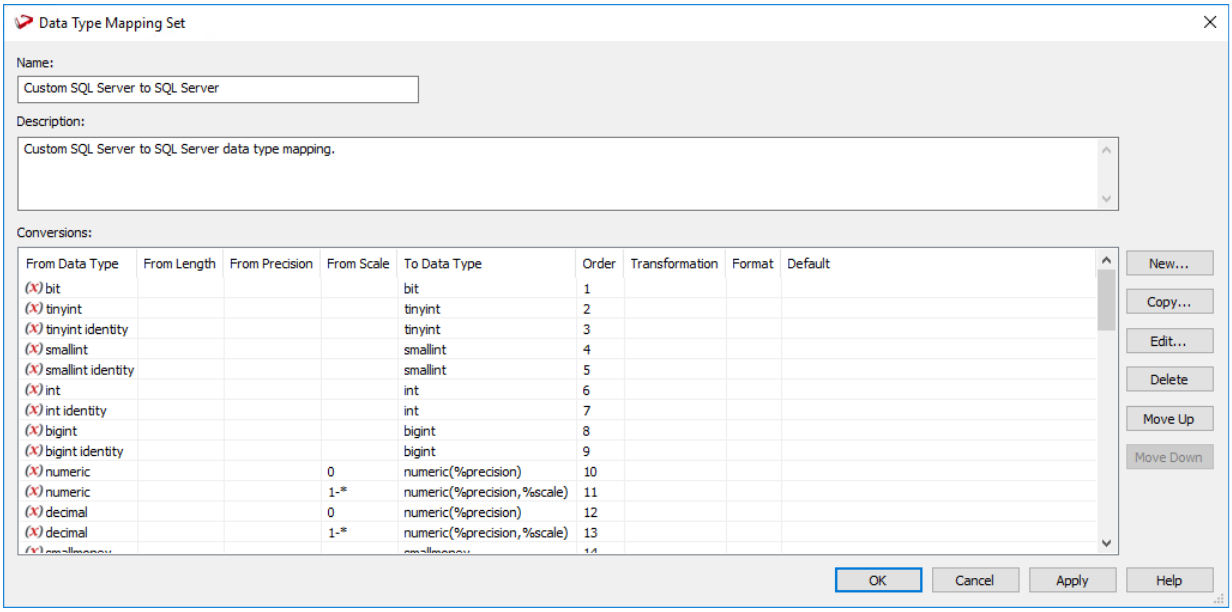


1. Select the Mapping Set to be copied from the list in the **Maintain Data Type Mapping Sets** window and then click the **Copy** button.



© Copyright, WhereScape Software Ltd All rights reserved.

2. Change the name and description of the new Mapping Set as required.

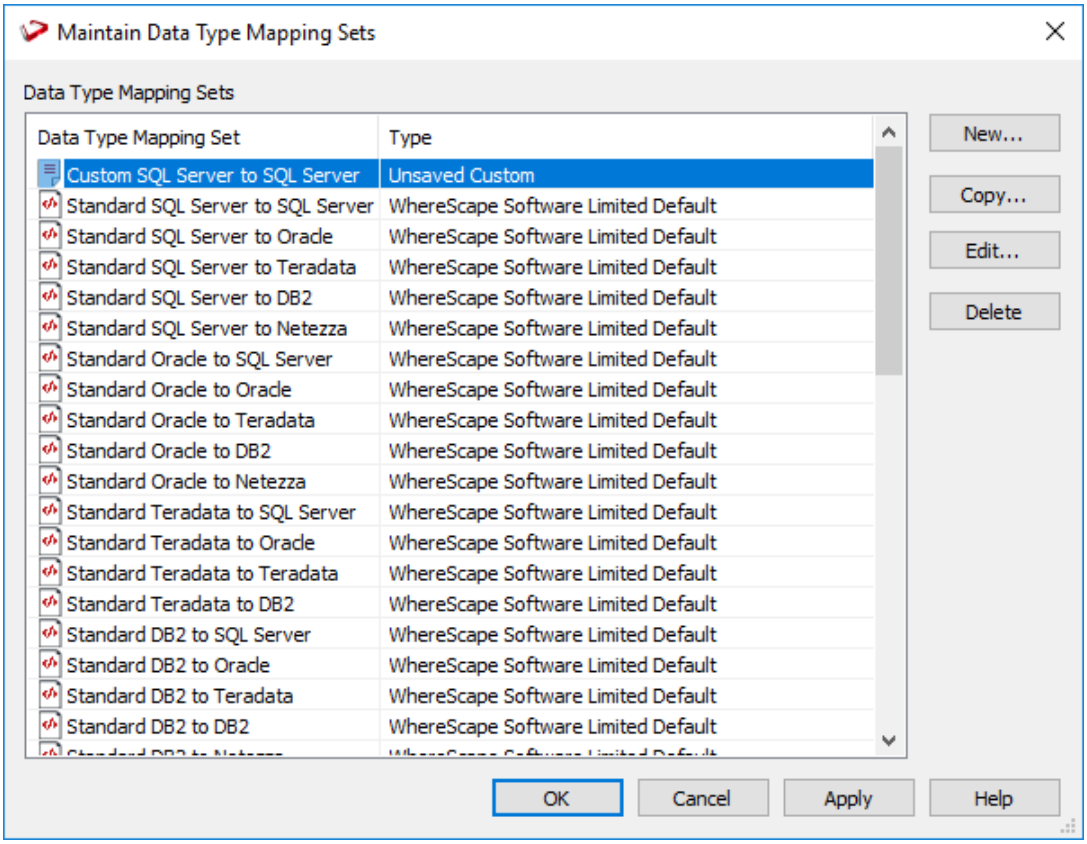


3. Use the buttons on the right side of the **Data Type Mapping Set** window to add new conversion parameters, edit, copy or delete a selected parameter.

Notes

To move a parameter up or down in the list, select it and then click the **Move Up** or **Move Down** button. The order of the data type mapping conversion parameters is significant because when loading a table, the procedure checks the data type mappings from top to bottom and stops when a data type and its parameters are correctly matched. A blank parameter means that it will match to anything.

4. Click **OK** when you're done. The new data type mapping set is listed in the **Maintain Data Mapping Sets** window.

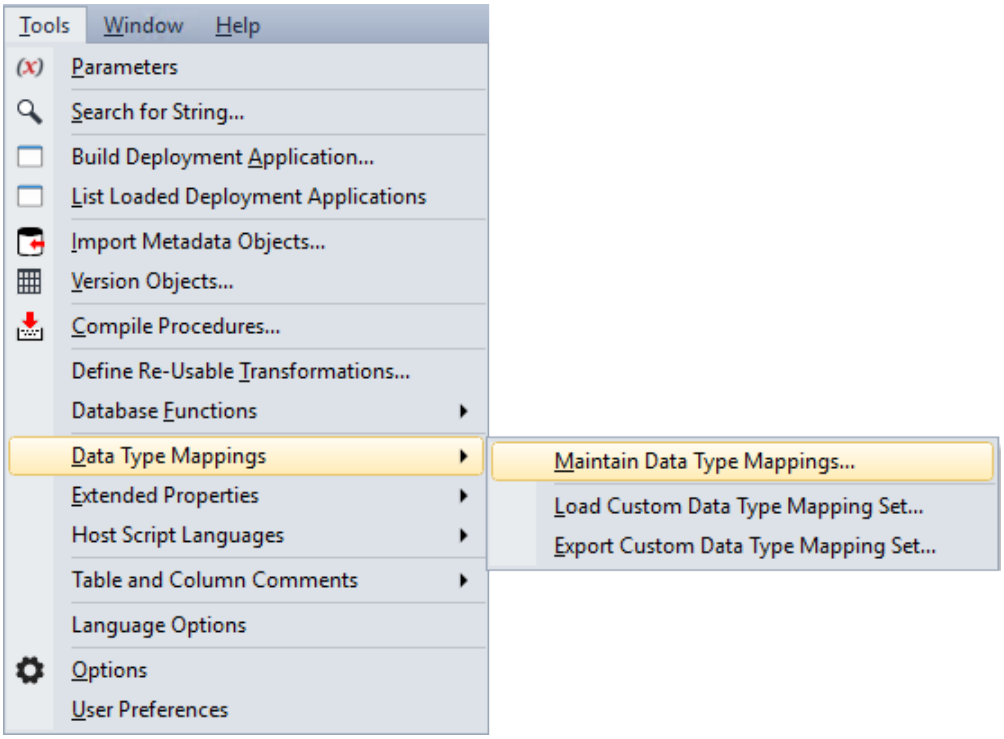


5. Click **OK** or **Apply** to save the new data type mapping set.

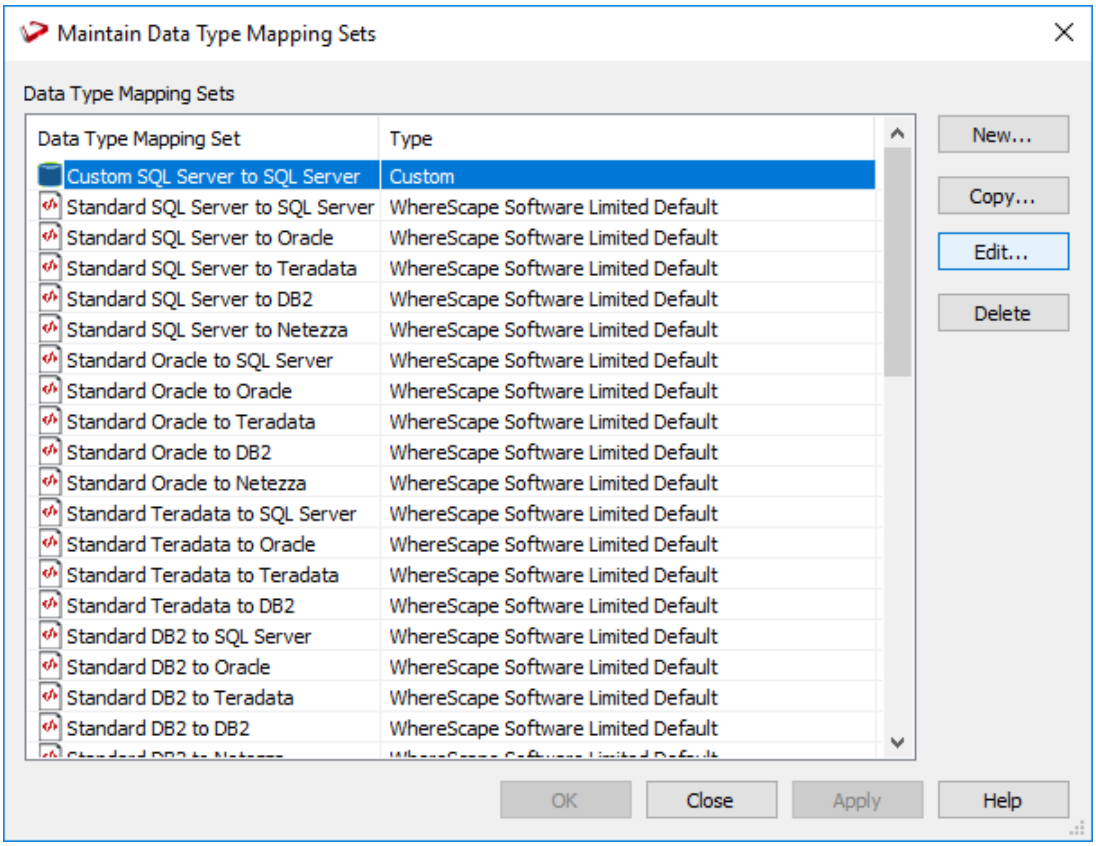
© Copyright, WhereScape Software Ltd All rights reserved.

Editing a Data Type Mapping Set

To edit a data type mapping set, select **Tools > Data Type Mappings > Maintain Data Type Mappings**.



1. Select the Mapping Set to be modified from the list in the **Maintain Data Type Mapping Sets** window and then click the **Edit** button.

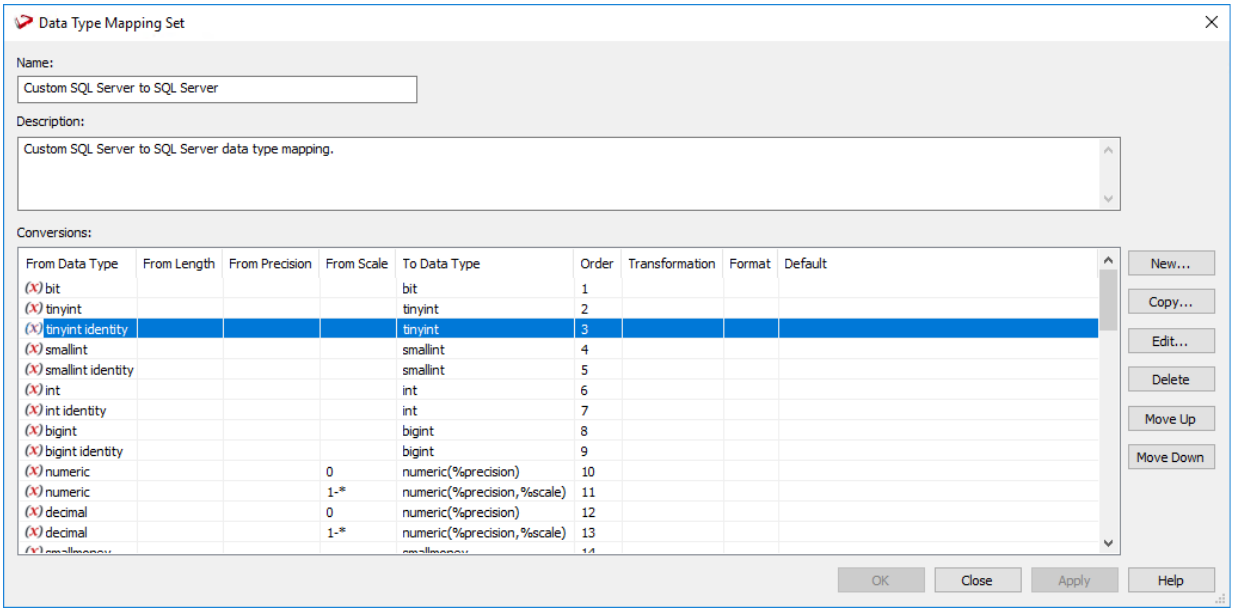


Notes

Standard Data Type Mapping sets are not editable, they can only be viewed.

© Copyright, WhereScape Software Ltd All rights reserved.

2. In the **Data Type Mapping Set** window, select the parameter you want to change from the **Conversions** pane. Use the scroll bar to see all the defined parameters.



3. Use the buttons on the right side of the **Data Type Mapping Set** window to add new conversion parameters, edit, copy or delete a selected parameter. These buttons are not available for standard mapping sets. Only **user defined** mapping sets are editable.

Notes

To move a parameter up or down in the list, select it and then click the **Move Up** or **Move Down** button. The order of the data type mapping conversion parameters is significant because when loading a table, the procedure checks the data type mappings from top to bottom and stops when a data type and its parameters are correctly matched. A blank parameter means that it will match to anything.

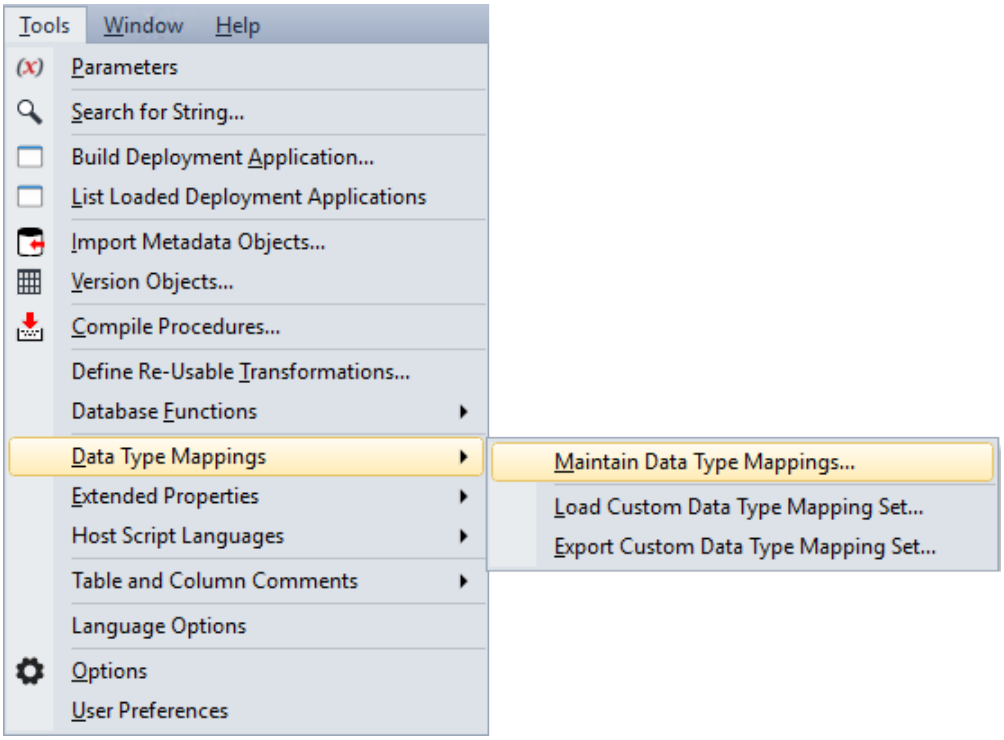
4. Click **OK** when you're done.

See previous sections for details on adding or copying a new data type mapping set.

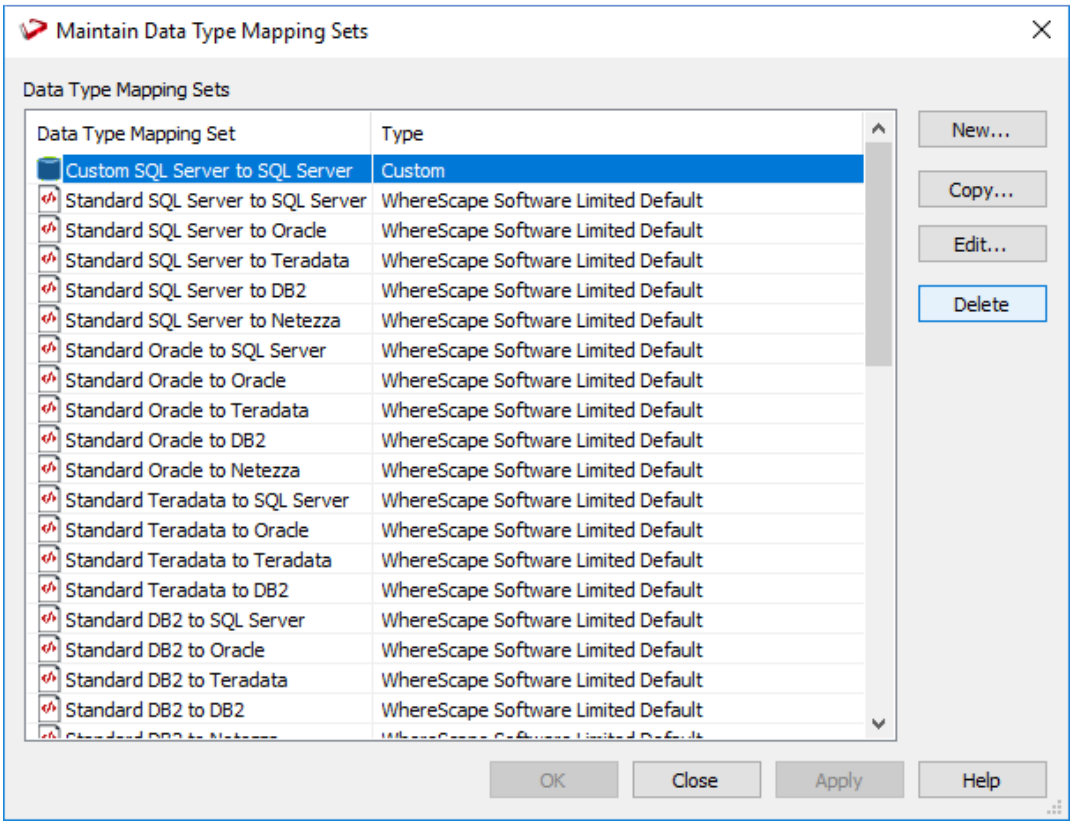
© Copyright, WhereScape Software Ltd All rights reserved.

Deleting a Data Type Mapping Set

1. To delete a data type mapping set, select **Tools > Data Type Mappings > Maintain Data Type Mappings**.



2. Select the Mapping Set to be deleted and then click the **Delete** button.

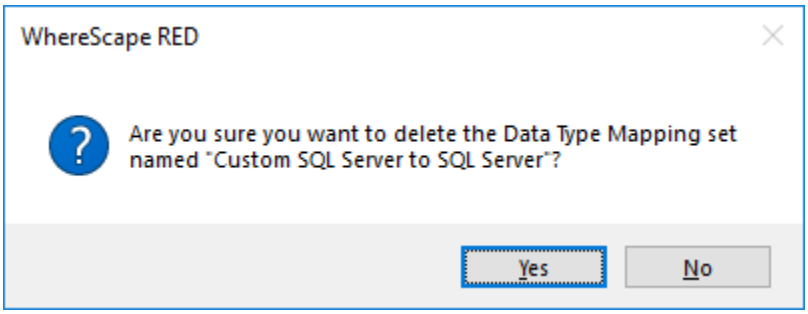


Notes

Standard data type mapping sets cannot be deleted. The **Delete** button is only enabled for non standard data type mapping sets.

© Copyright, WhereScape Software Ltd All rights reserved.

3. A prompt is displayed asking you to confirm the action. Click **Yes** to proceed or **No** to cancel.

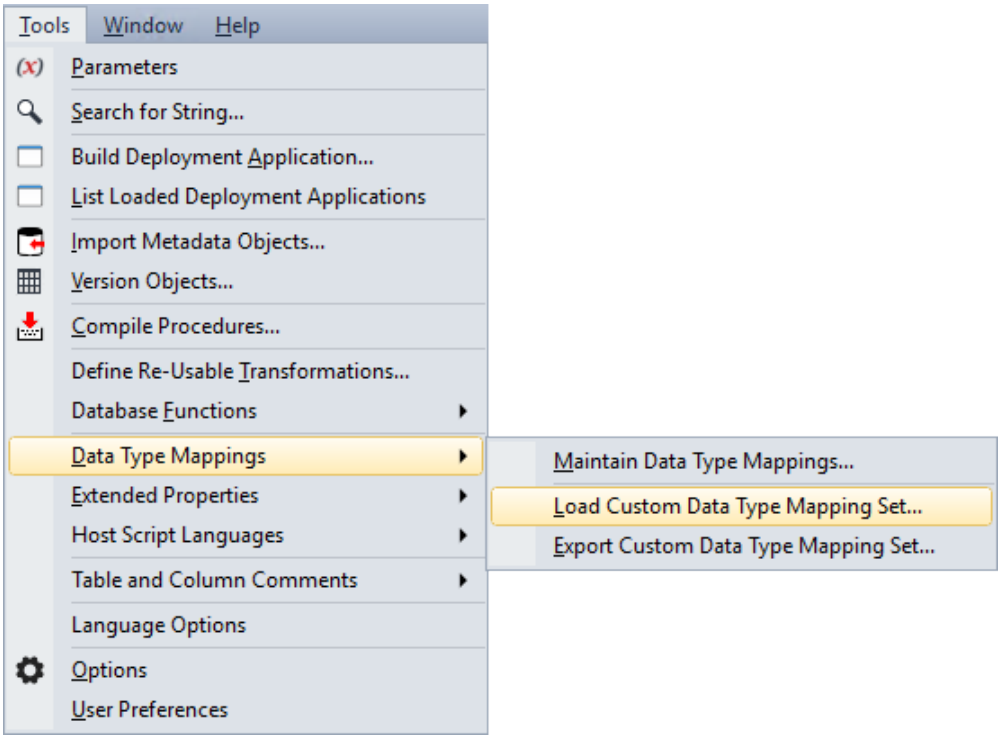


© Copyright, WhereScape Software Ltd All rights reserved.

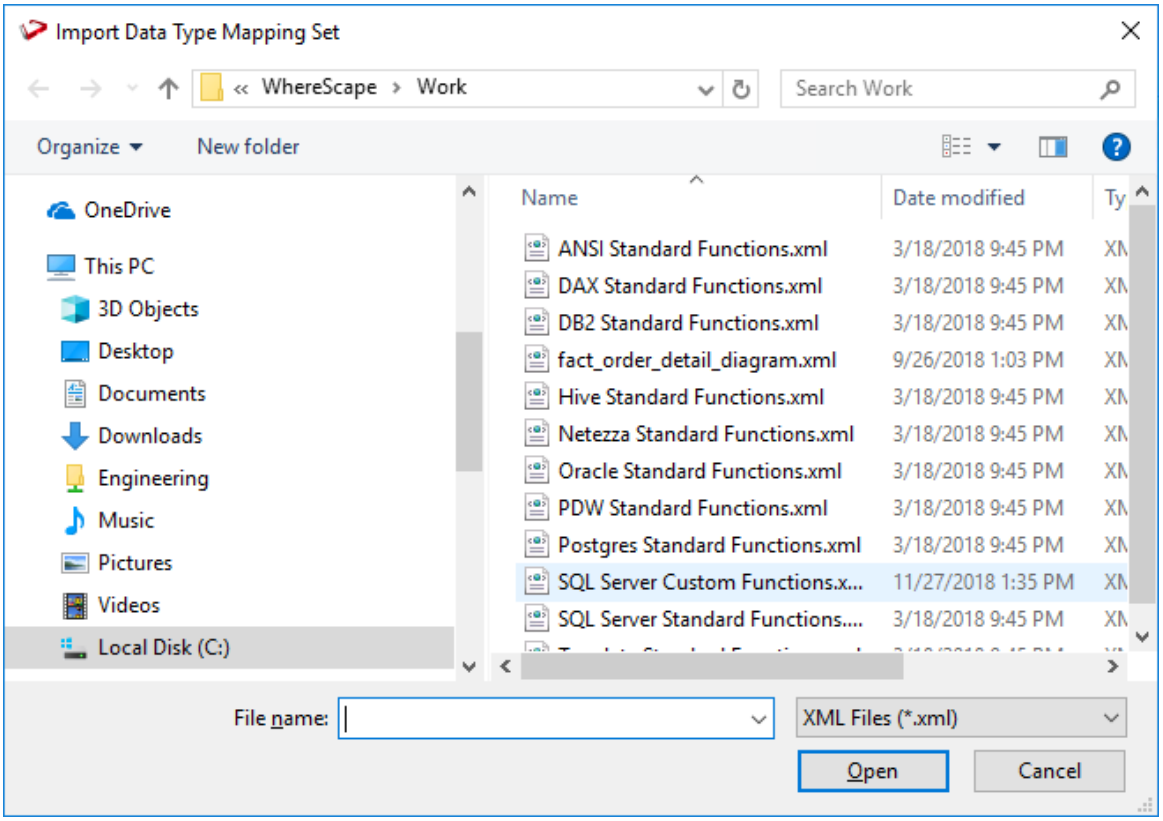
Loading Custom Data Type Mapping Sets

The **Load Custom Data Type Mapping Set** menu option enables you to load a custom data type mapping set from an XML file into the RED metadata repository.

To load a data type mapping set, select **Tools > Data Type Mappings > Load Custom Data Type Mapping Set**.



The following window is displayed. Select the xml file to load the data type mapping set. By default, RED expects the xml files to be in the **ProgramData\WhereScape\Work** directory.

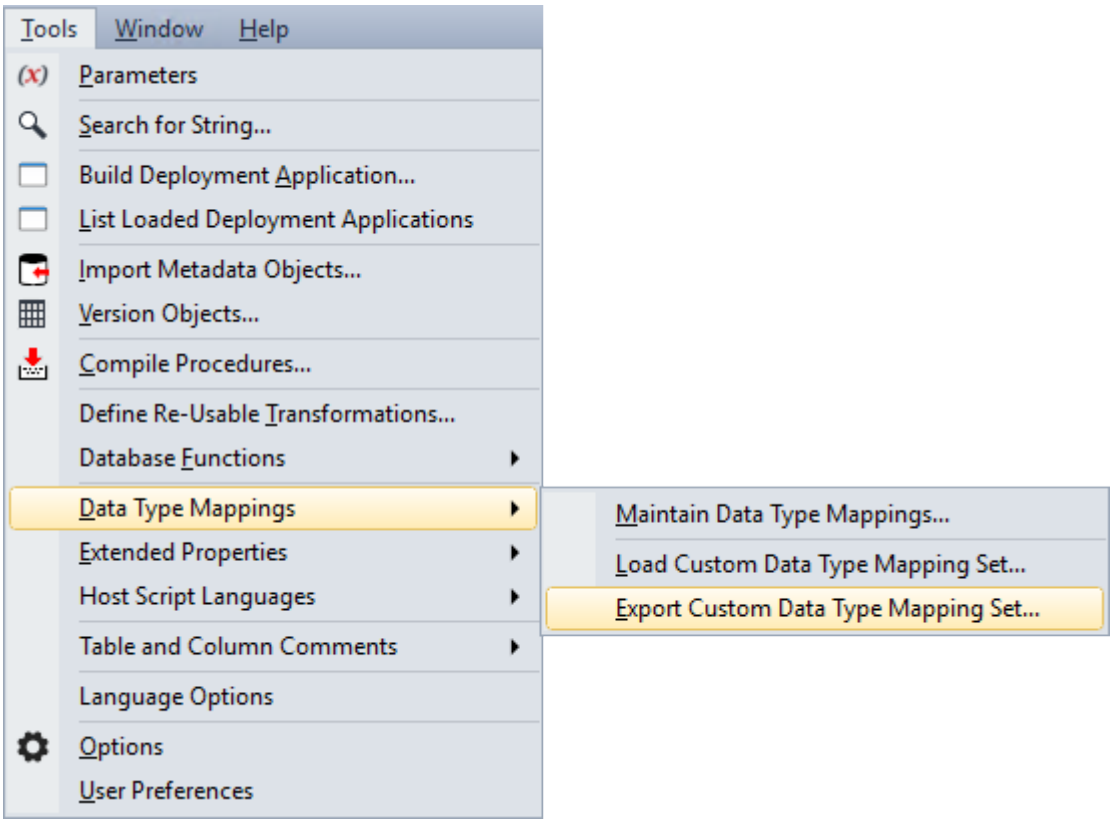


© Copyright, WhereScape Software Ltd All rights reserved.

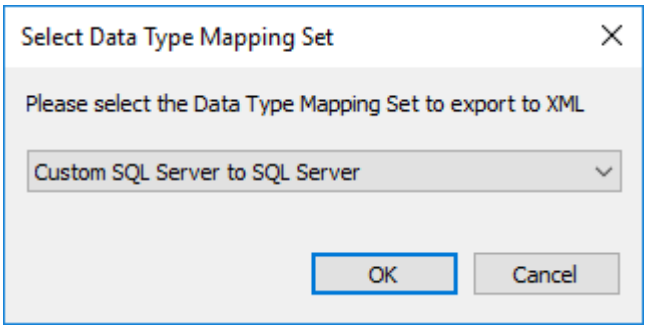
Exporting Custom Data Type Mapping Sets

The **Export Custom Data Type Mapping Set** menu option enables you to export a custom data type mapping set from the RED metadata repository to an XML file.

To export a data type mapping set, select **Tools > Data Type Mappings > Export Custom Data Type Mapping Set**.

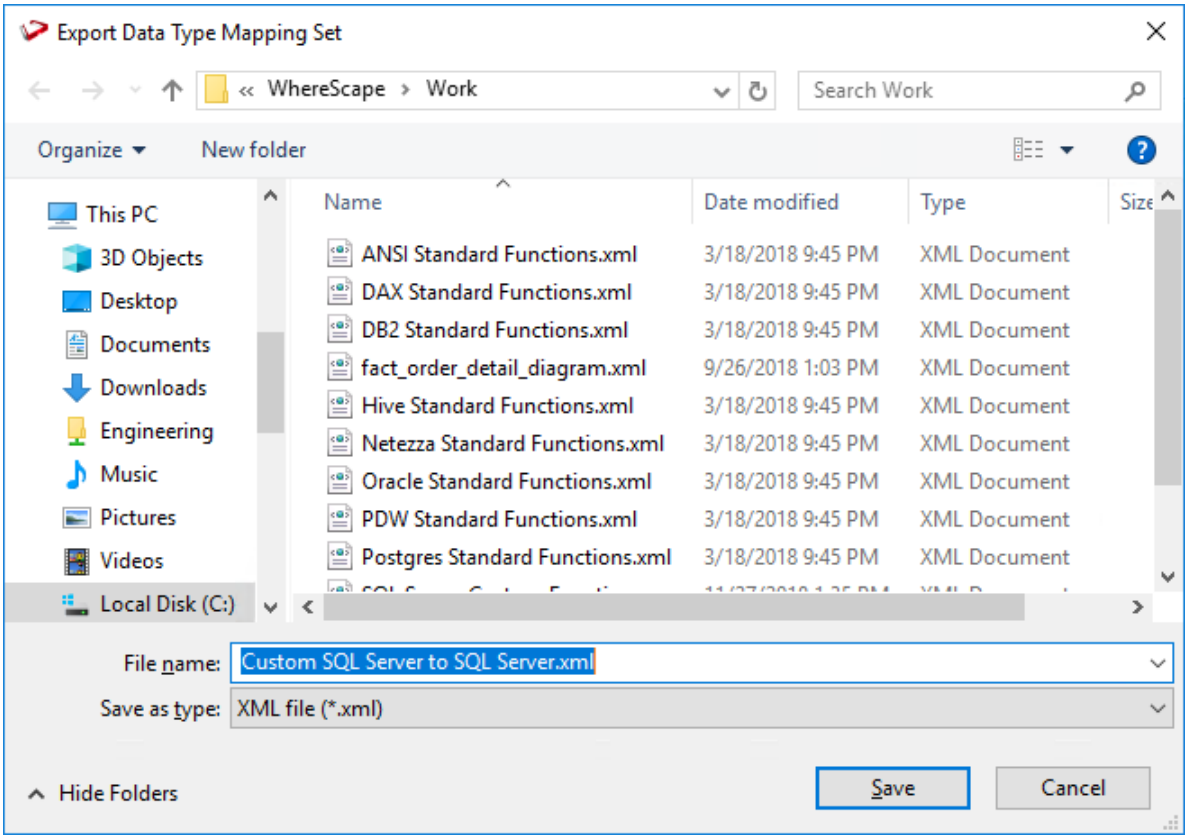


Select the data type mapping set to export from the drop-down list. Click **OK** to proceed.



© Copyright, WhereScape Software Ltd All rights reserved.

By default, RED exports the xml file to the **ProgramData\WhereScape\Work** directory, but this can be changed. Change the File name if necessary and click **Save**.



© Copyright, WhereScape Software Ltd All rights reserved.

Custom Data Type Mapping Set Examples

WhereScape RED enables you to create **Custom** Data Type Mapping Sets. This gives you the ability to automatically change the data type of any column or to add column transformations, when dragging and dropping new Load tables.

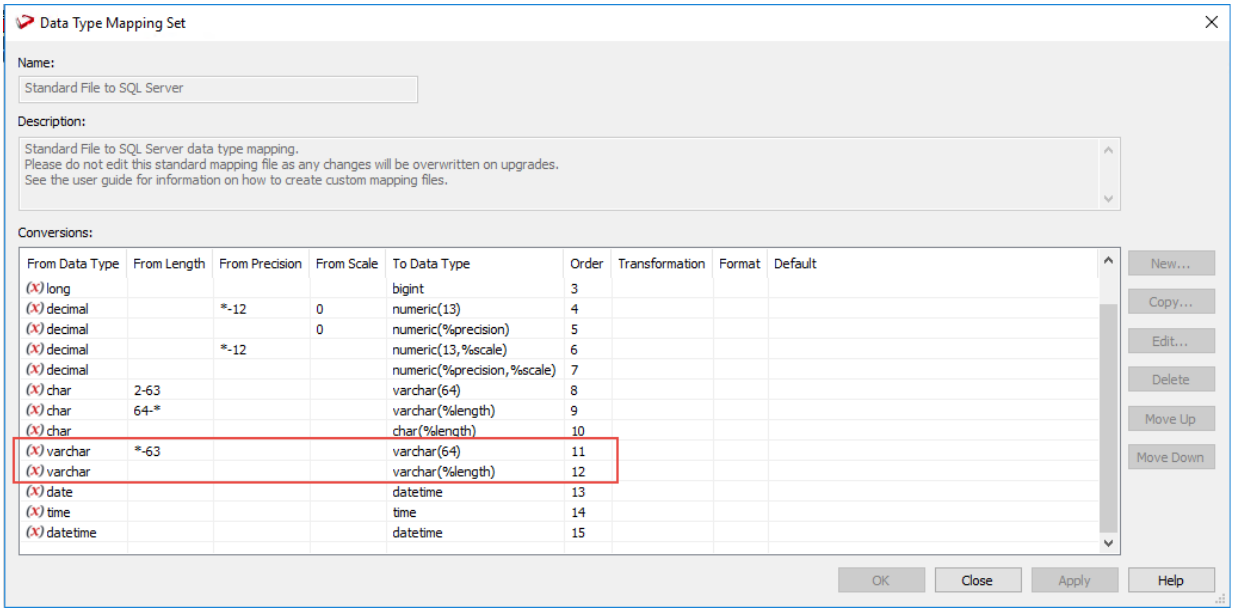
The examples in this topic demonstrate how **Custom** Data Type Mapping Sets can be configured, using the following variables:

- %length
- %scale
- %precision
- %table_name
- %column_name
- %format
- %prompt

%length

In the example below, when converting a varchar in a standard file to SQL Server format, we follow the following steps in the given order:

- If the varchar is of a length less than or equal to 63, the **data type** will become varchar(64).
- If the first step was NOT applied, i.e. the varchar is of a length greater than 63, then the **data type** will become varchar(%length); where we substitute the length for the variable '%length'. Thus if the varchar is of length 64 then the resulting data type will be varchar(64), but if the varchar is of length 123 then the resulting data type will be varchar(123).

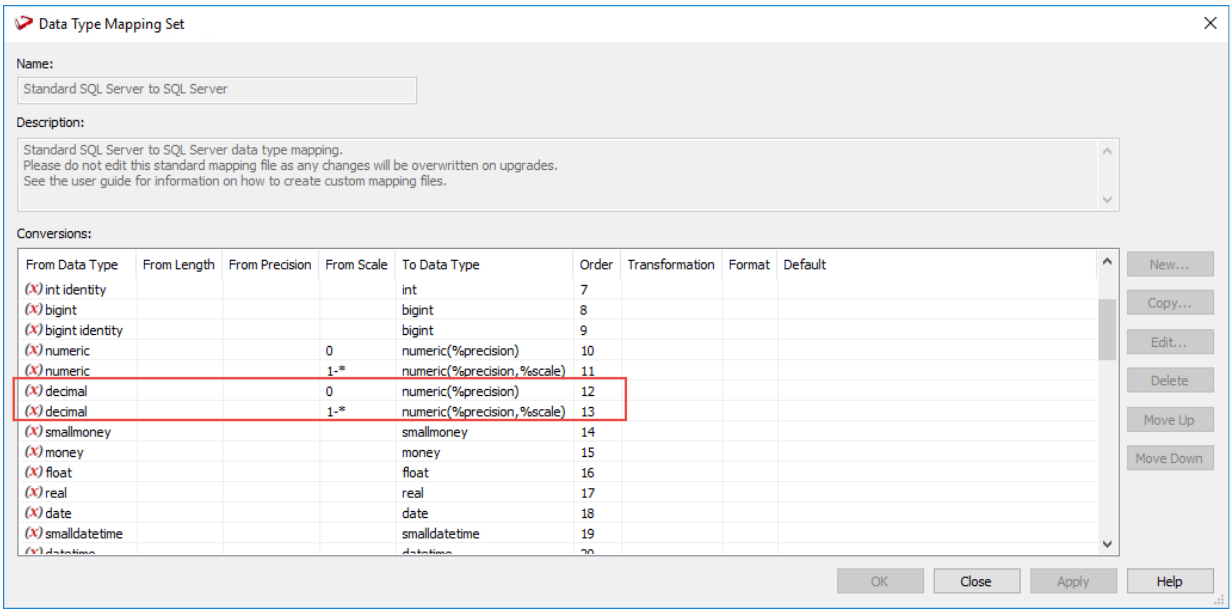


%scale

In the example below, when converting a decimal in SQL Server to SQL Server format, the following steps are done in the given order:

- If the decimal has a scale of zero, the **data type** will become numeric(%precision); where we substitute the number of digits in the number for the variable '%precision'. Thus if the decimal has 8 digits then the resulting data type will be numeric(8).
- If the first step was NOT applied, i.e. the decimal has a scale of 1 or greater, then the **data type** will become numeric(%precision,%scale); where we not only substitute the number of digits in the number for the variable '%precision', but we also substitute the scale for the variable '%scale'. Thus, if the decimal is made up of 8 digits and has 3 digits after the decimal point (example 12345,678), the resulting data type will be numeric(8,3).

© Copyright, WhereScape Software Ltd All rights reserved.



Notes

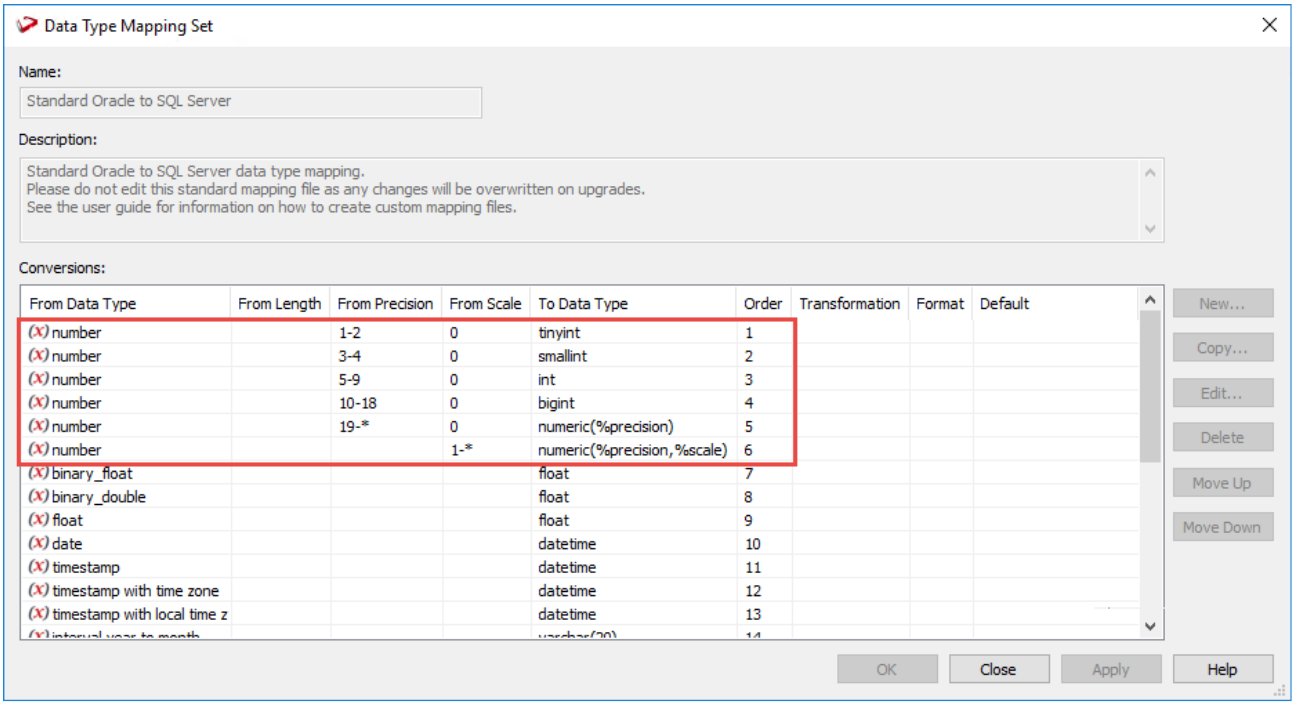
The **Scale>** is the number of digits to the right of the decimal point in a number.

© Copyright, WhereScape Software Ltd All rights reserved.

%precision

In the example below, when converting a number in Oracle to SQL Server format, the following steps are done in the given order:

- If the number is 1 or 2 digits long with a scale of zero, the **data type** will become tinyint.
- If the number is 3 or 4 digits long with a scale of zero, the **data type** will become smallint.
- If the number is 5-9 digits long with a scale of zero, the **data type** will become int.
- If the number is 10-18 digits long with a scale of zero, the **data type** will become bigint.
- If the number is 19 or greater digits long with a scale of zero, the data type will become numeric(%precision); where we substitute the number of digits in the number for the variable '%precision'. Thus if the number is made up of 22 digits, the resulting data type will be numeric(22).
- If however, the scale is not zero and there are digits to the right of the decimal point in the number, then the **data type** will become numeric(%precision,%scale); where we substitute both the number of digits in the number for the variable '%precision' and the number of digits to the right of the decimal point for the variable '%scale'. Thus if the number is made up of 8 digits and has 3 digits after the decimal point (example 12345,678), the resulting data type will be numeric(8,3).



Notes

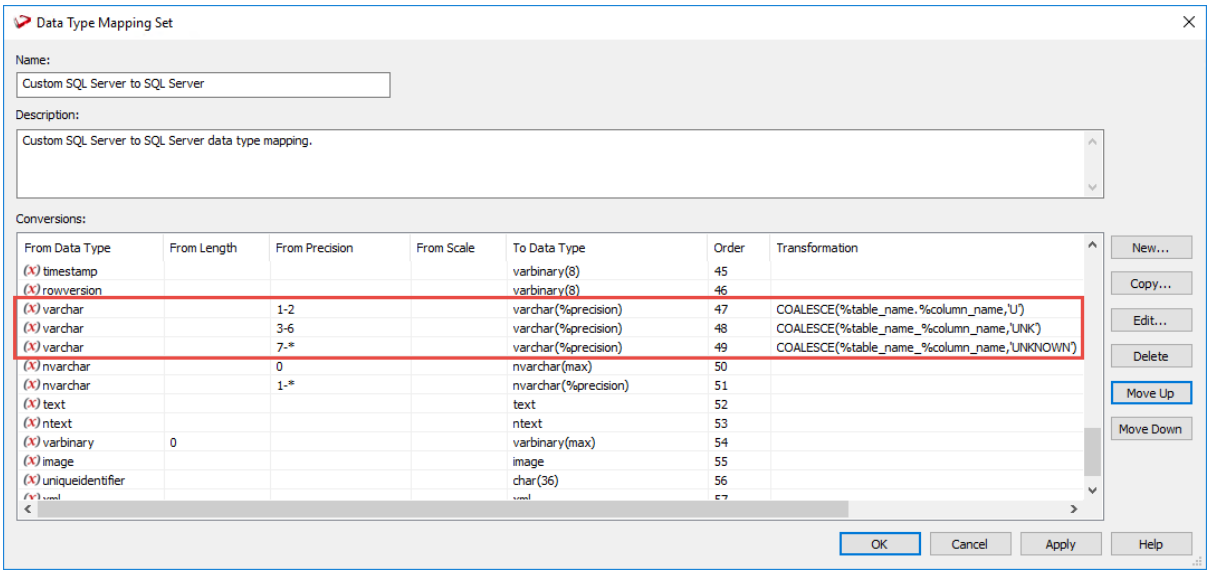
The **Precision** is the total number of digits in a number.

%table_name and %column_name

In the example below, the following transformations are used to handle NULL for different lengths of varchars:

- If the varchar is 1 or 2 digits/chars long, the **data type** will become varchar(%precision); where we substitute the number of digits/chars in the varchar for the variable '%precision'. Secondly, the **value** of the column will become the column value (if it is not null), else it will become 'U'.
- If the varchar is 3-6 digits/chars long, the **data type** will become varchar(%precision); where we substitute the number of digits/chars in the varchar for the variable '%precision'. Secondly, the **value** of the column will become the column value (if it is not null), else it will become 'UNK'.
- If the varchar is 7 or more digits/chars long, the **data type** will become varchar(%precision); where we substitute the number of digits/chars in the varchar for the variable '%precision'. Secondly, the **value** of the column will become the column value (if it is not null), else it will become 'UNKNOWN'.

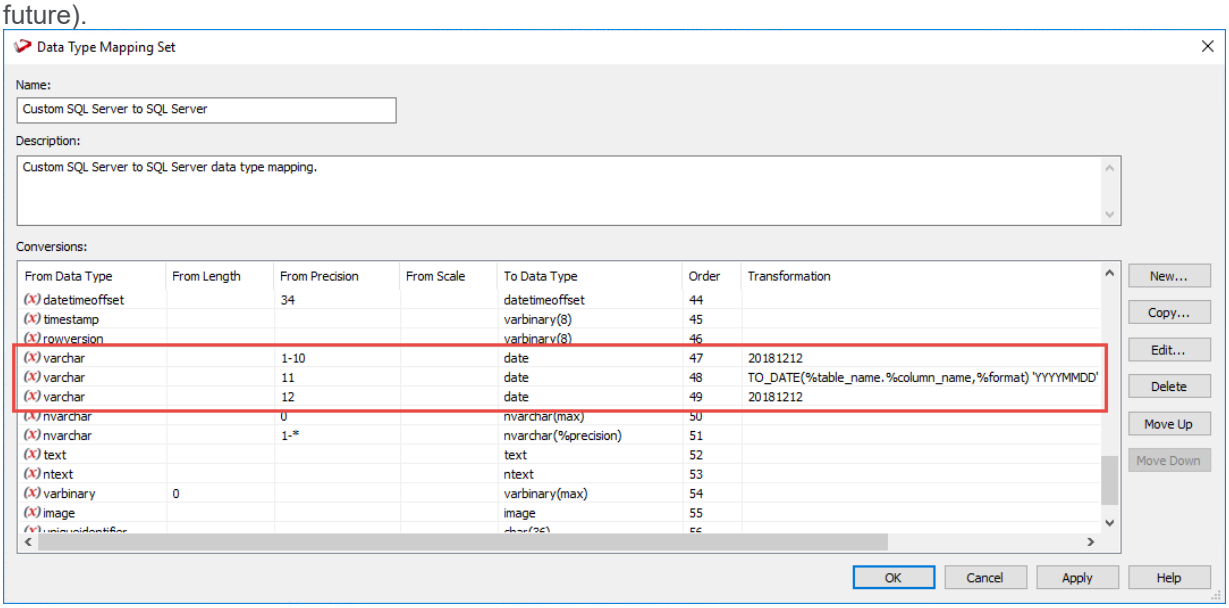
© Copyright, WhereScape Software Ltd All rights reserved.



%format

In the example below, the following transformations are used to convert a certain character field to a date:

- If the varchar has a length of 1-10, the **data type** will become date and the **value** of the column will become the date 20131212 (a chosen date in the future).
- If the varchar has a length of 11, the **data type** will become date and the **value** of the column will use the transformation TO_DATE(%table_name.%column_name,%format); where we substitute 'YYYYMMDD' for the variable '%format'. Thus the value of the column will be converted to a date of format 'YYYYMMDD'.
- If the varchar has a length of 12 or greater, the **data type** will become date and the **value** of the column will become the date 20181212 (a chosen date in the future).

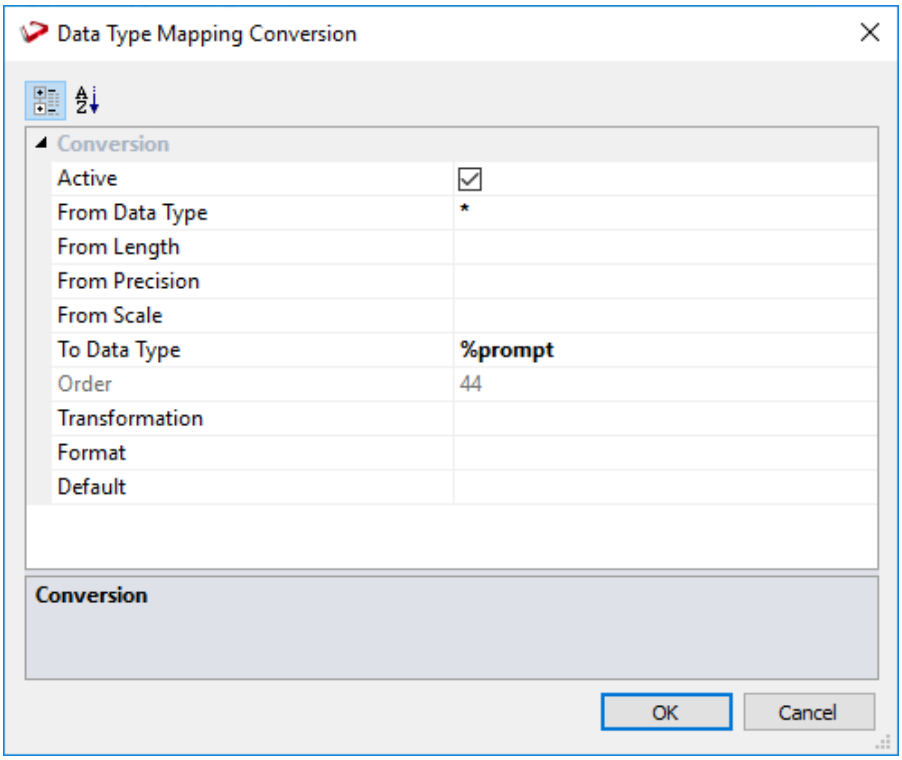


%prompt

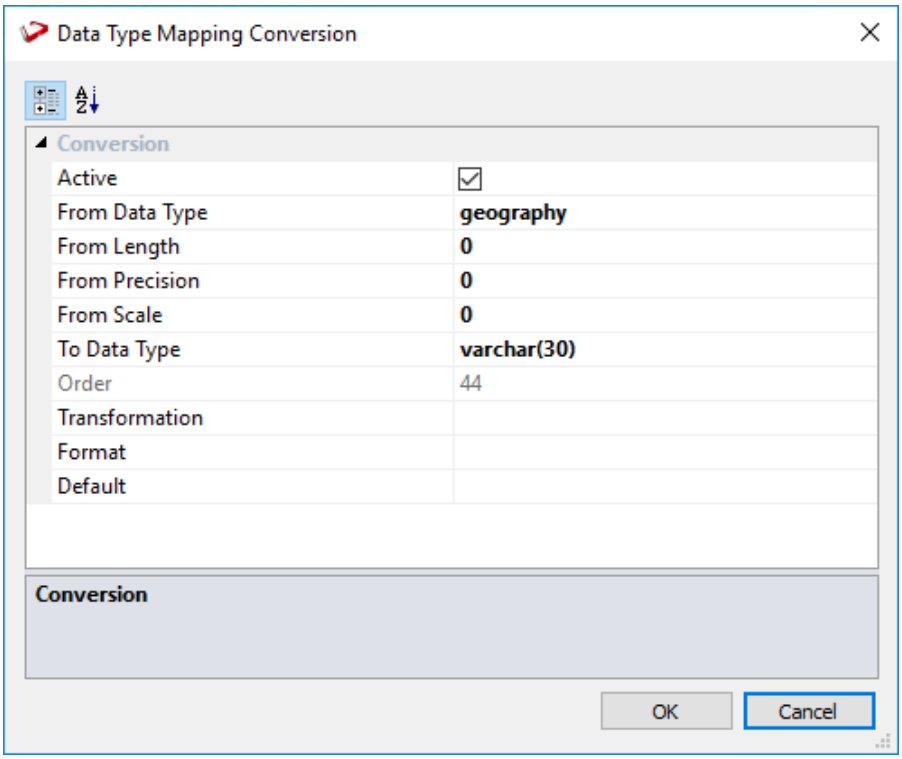
In the example below, %prompt is used to help the user define a mapping for an unknown data type that is not already mapped in the previous mapping rules.
This variable must be used with a **custom** Data Type mapping set, as described in the following steps:

- Create a new custom set or copy from an existing set.
- Create a new Data Type mapping with a **From Data Type** of star (*) and a **To Data Type** of **%prompt**. Click **OK** to save the New Data Type Mapping to the Custom set.

© Copyright, WhereScape Software Ltd All rights reserved.

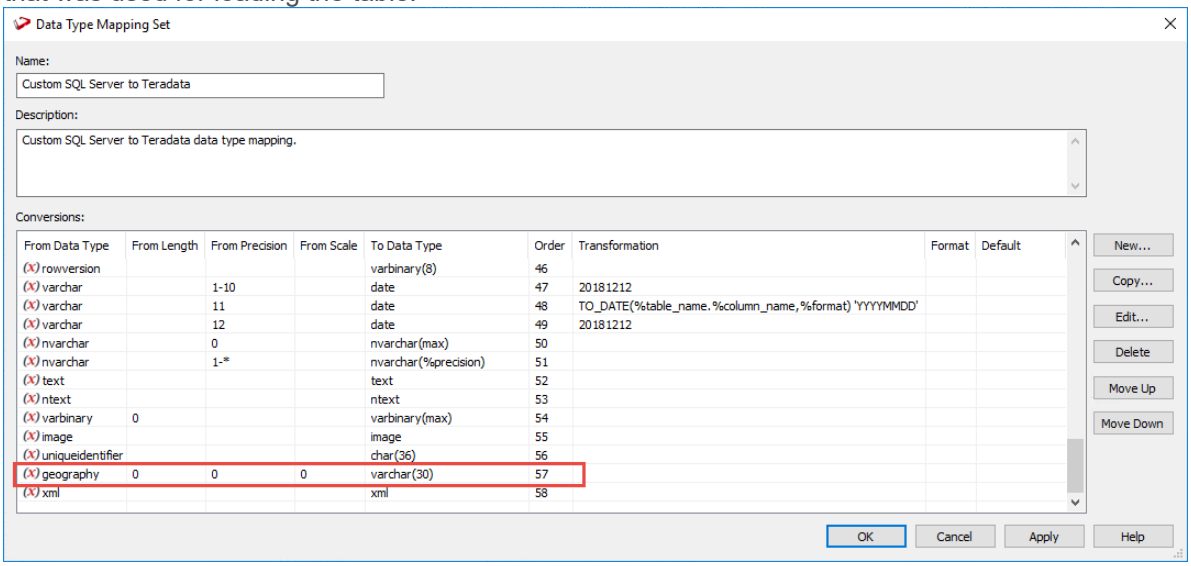


- When browsing a connection to Load a table, set the **Data Type Mapping Set** to the new **Custom SQL Server to Teradata** mapping set. This can be set in the **List Source Tables Connection** window or in the connection **Properties** window. Refer to **Browsing a Connection** for details.
- As the table is dragged and dropped to the middle pane, RED prompts to have the new data type mapping defined.
- In the example below, just before loading the table, users can map the unknown **geography** SQL Server data type mapping to a **varchar(30)** in Teradata.



© Copyright, WhereScape Software Ltd All rights reserved.

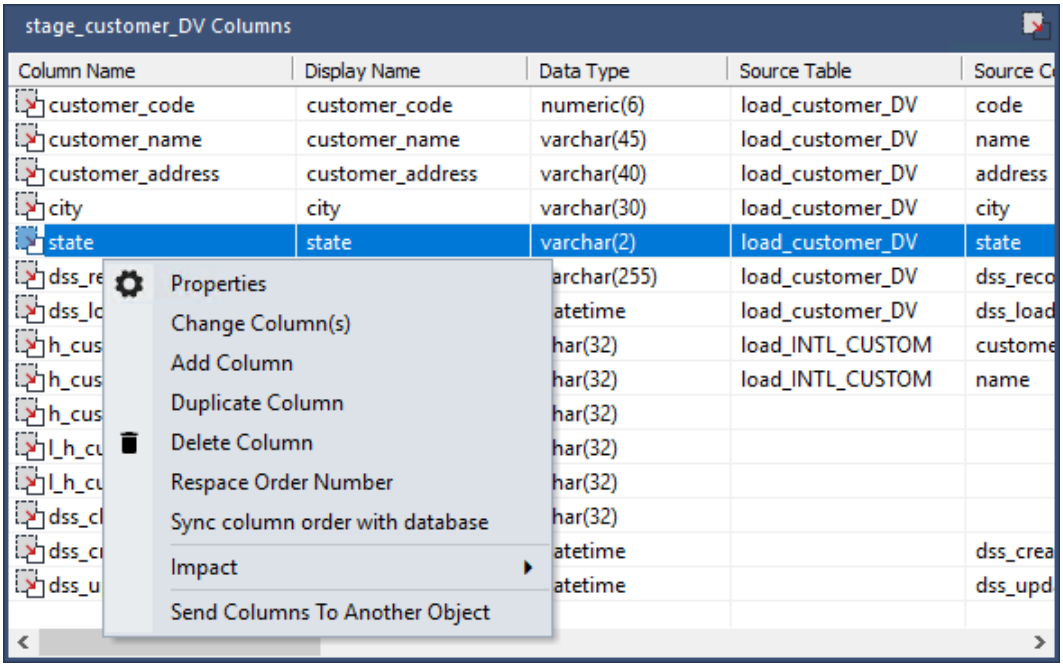
- Clicking **OK** or **Apply** in the **Data Type Mapping Set** window, saves the parameter to the custom set that was used for loading the table.



© Copyright, WhereScape Software Ltd All rights reserved.

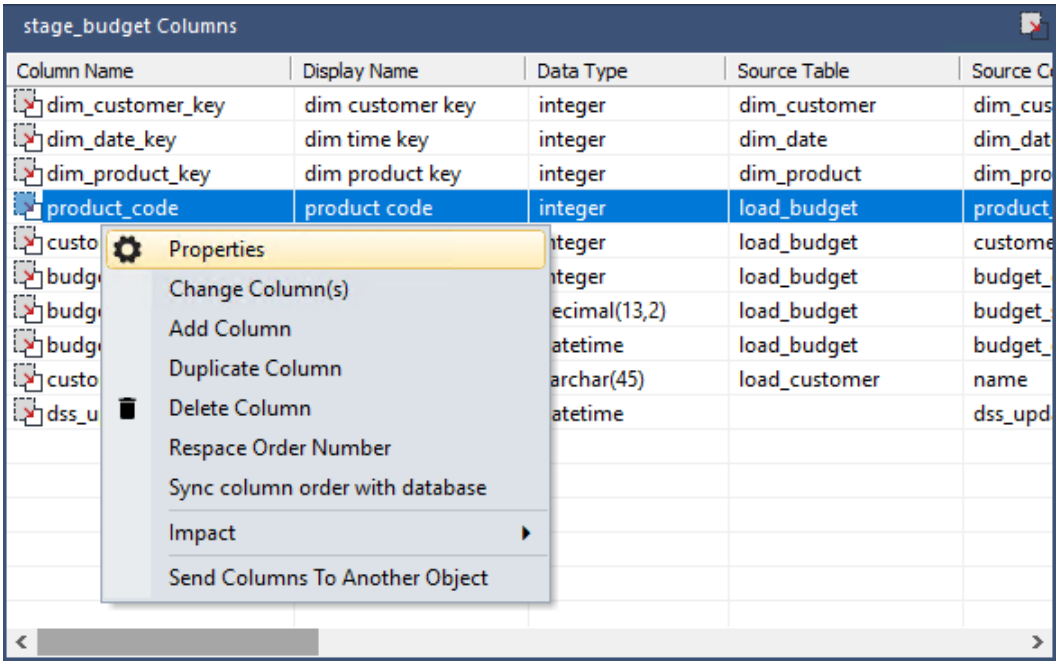
Column Context Menu

To view the column context menu, click on an object in the left pane to display the columns in the middle pane. When positioned on a column in the middle pane, right-click on the column to bring up the menu.



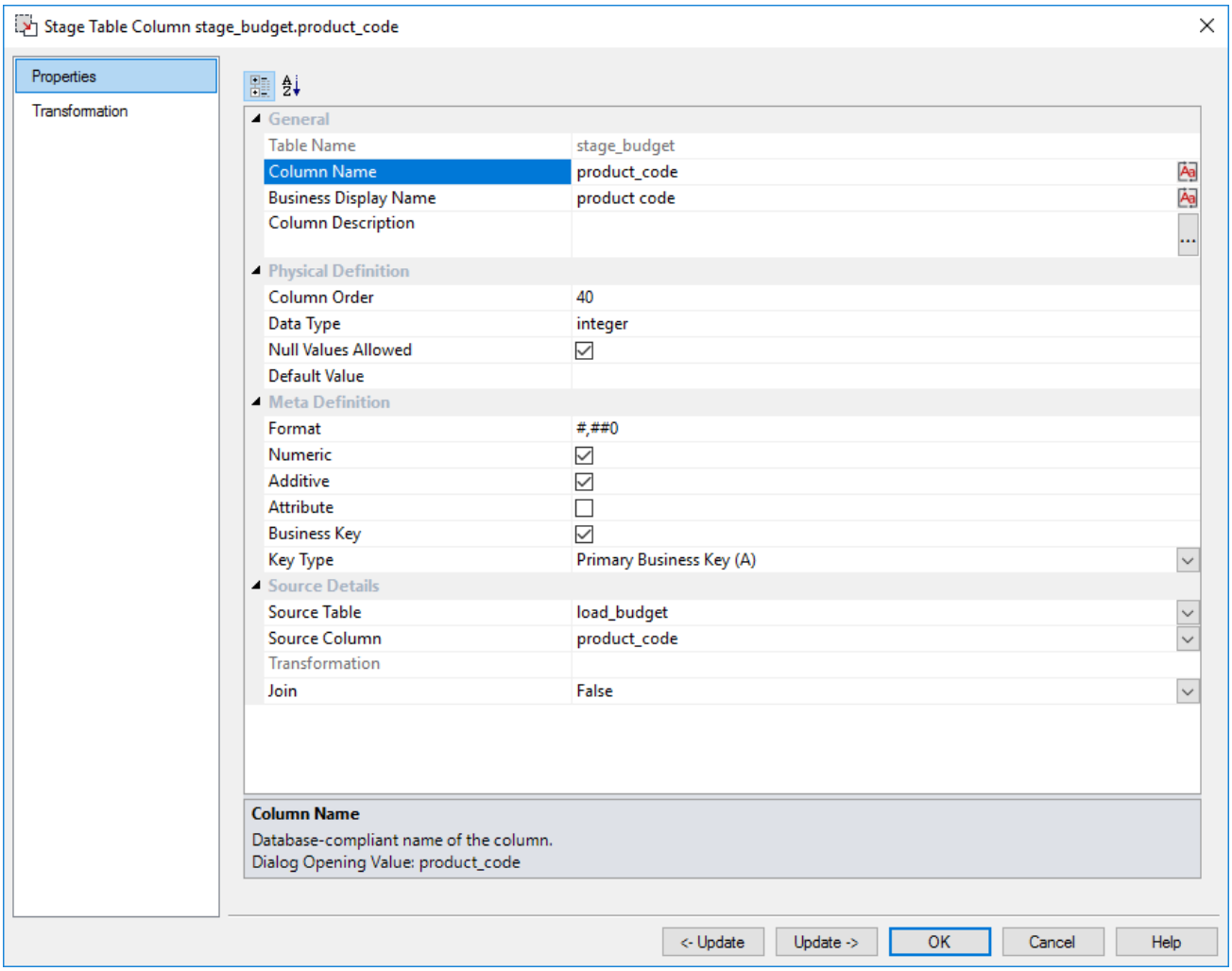
Properties

To display the column Properties, right-click on a column in the middle pane and select **Properties**.



© Copyright, WhereScape Software Ltd All rights reserved.

Edit any field as required and then click **OK** to close.

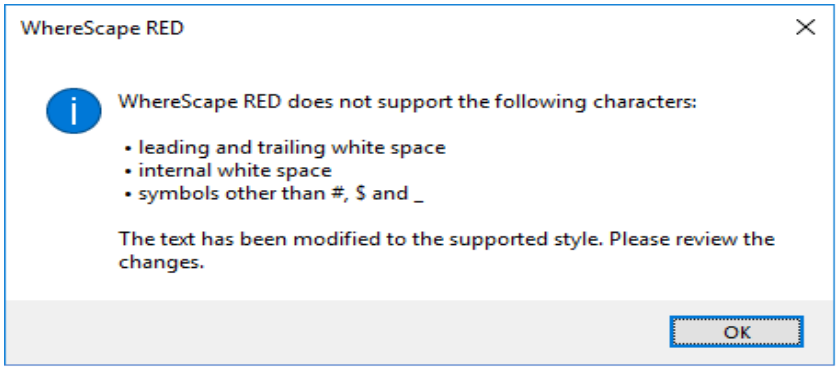


Warning

WhereScape RED does not support the following characters in **Column Names**:

- leading and trailing white spaces
- internal white spaces
- symbols other than #, $ and _

If users attempt to enter any of the above characters in Column Names, the following message is displayed, advising users to review changes made by RED to correct any unsupported column name characters:



© Copyright, WhereScape Software Ltd All rights reserved.

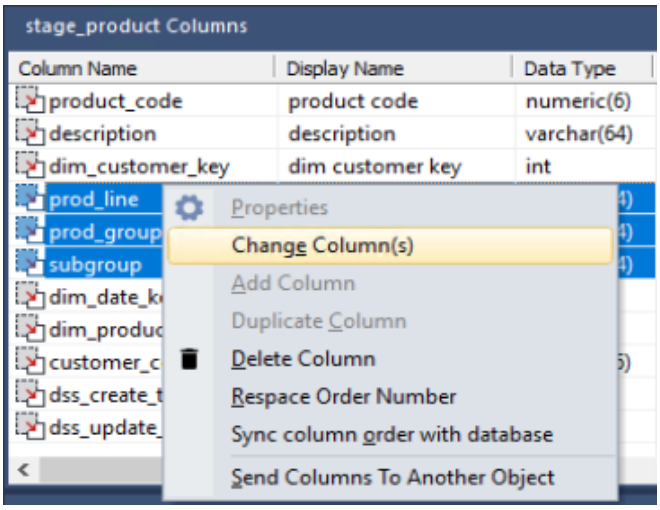
Notes:

There is a variation in column Properties, depending on the object type.

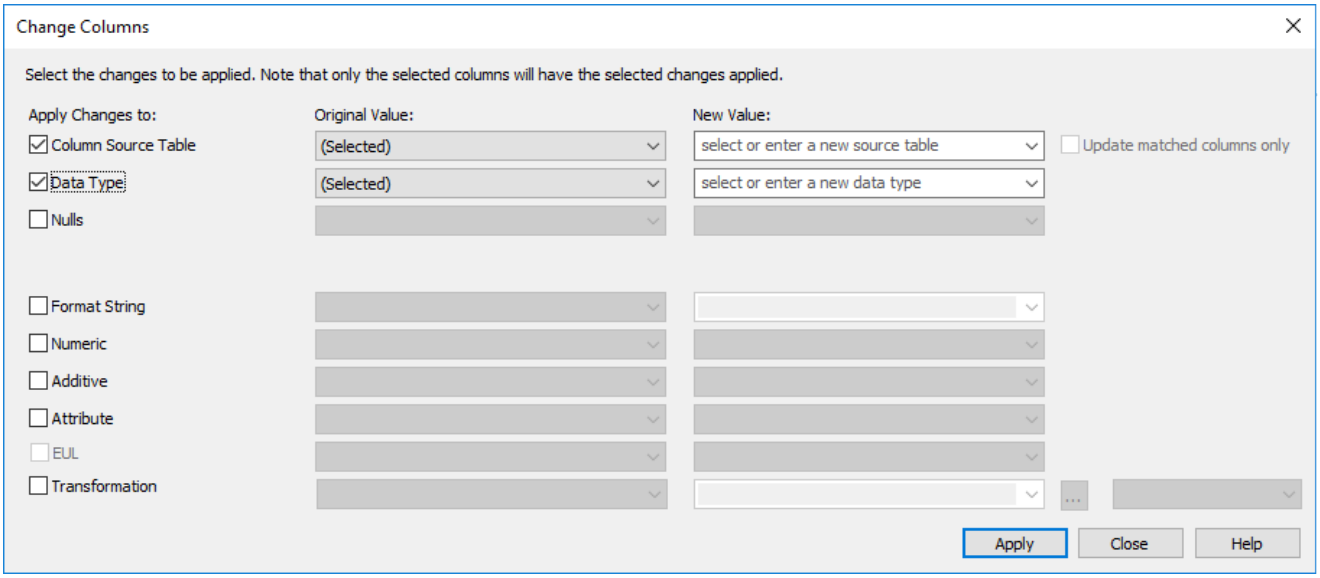
- For Dimension tables, refer to **Dimension Column Properties**
- For Stage tables, refer to **Stage Table Column Properties**
- For Data Store tables, refer to **Data Store Column Properties**
- For EDW 3NF tables, refer to **EDW 3NF Table Column Properties**
- For Fact tables, refer to **Fact Table Column Properties**

Change Column(s)

To change the properties of multiple columns, right-click the columns in the middle pane and select **Change Column(s)**.

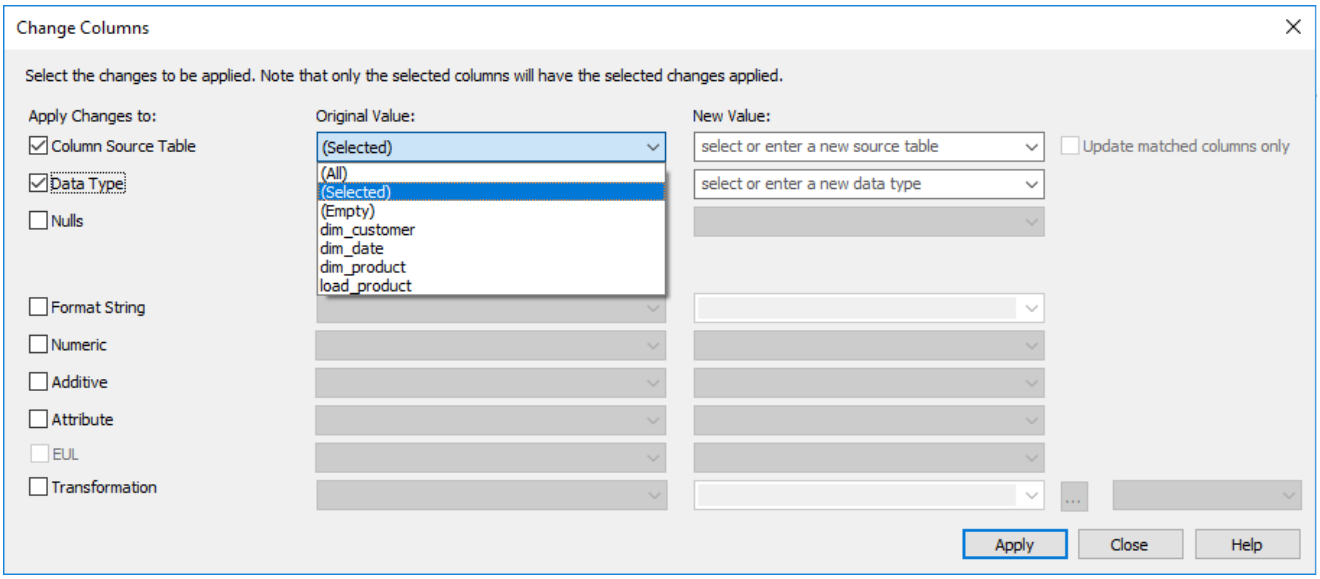


Select the relevant **check boxes** in the left pane to specify the properties to be changed. Selecting a check box enables you to change the value for that field in the column properties.



© Copyright, WhereScape Software Ltd All rights reserved.

In the **Original Value** column, select the value(s) to be changed.

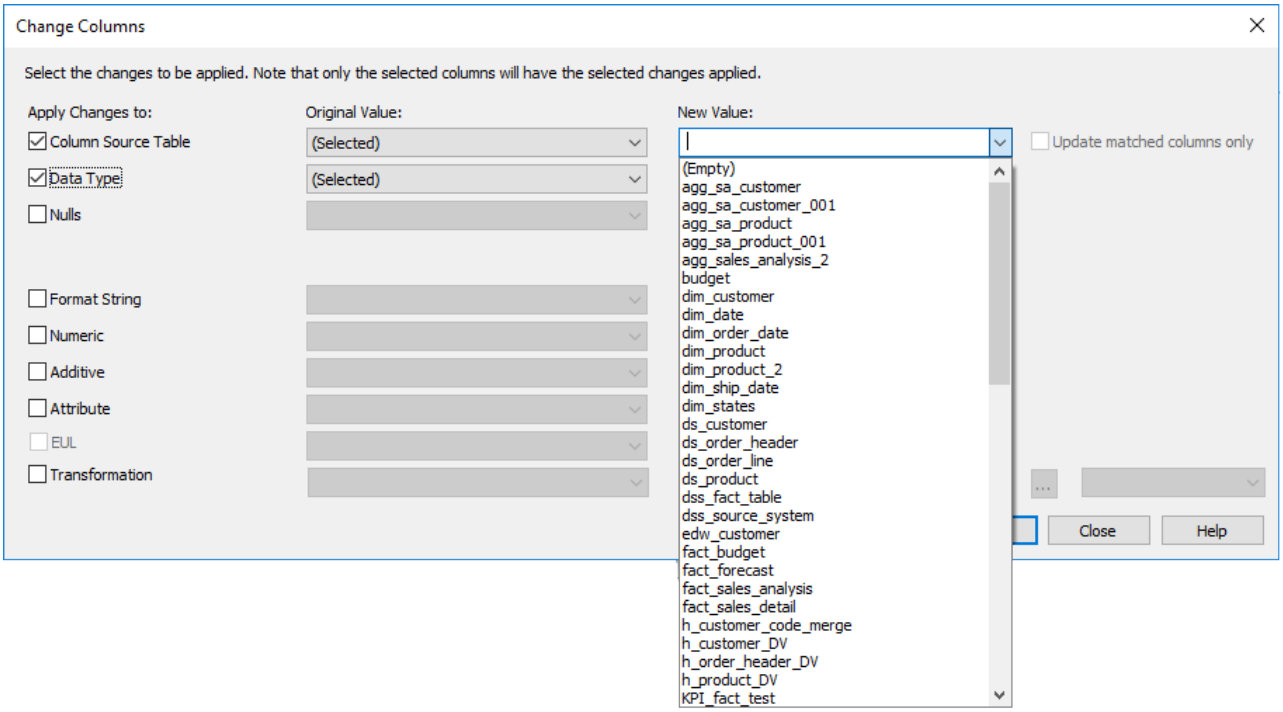


- Choosing **(All)** changes the selected property for all of the columns in the table.
- Choosing **(Selected)** changes the selected property for the selected column in the table.
- Choosing **(Empty)** changes the selected property for all of the columns where that property field is empty. This option is only available if there is a column where this property is empty.
- Choosing one of the **other** options changes the selected property for all the columns in the table having that value.

Note:

(Selected) is the default for the **Original Value** column.

In the **New Value** column, select the new value to be assigned; or key in the new value.



Notes:

- When editing the **Column Source Table** property, selecting the **Update matched columns only** check-box validates that the selected column name exists in the new source table. If it does not exist,

© Copyright, WhereScape Software Ltd All rights reserved.

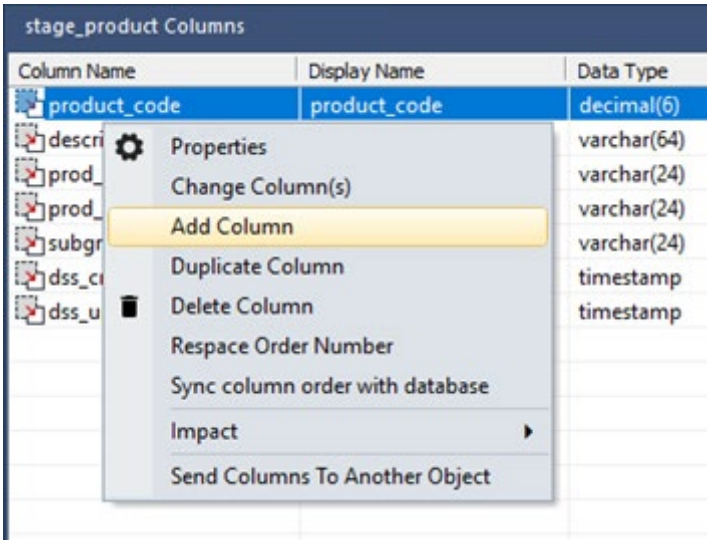
Notes:

then the update does not take place.

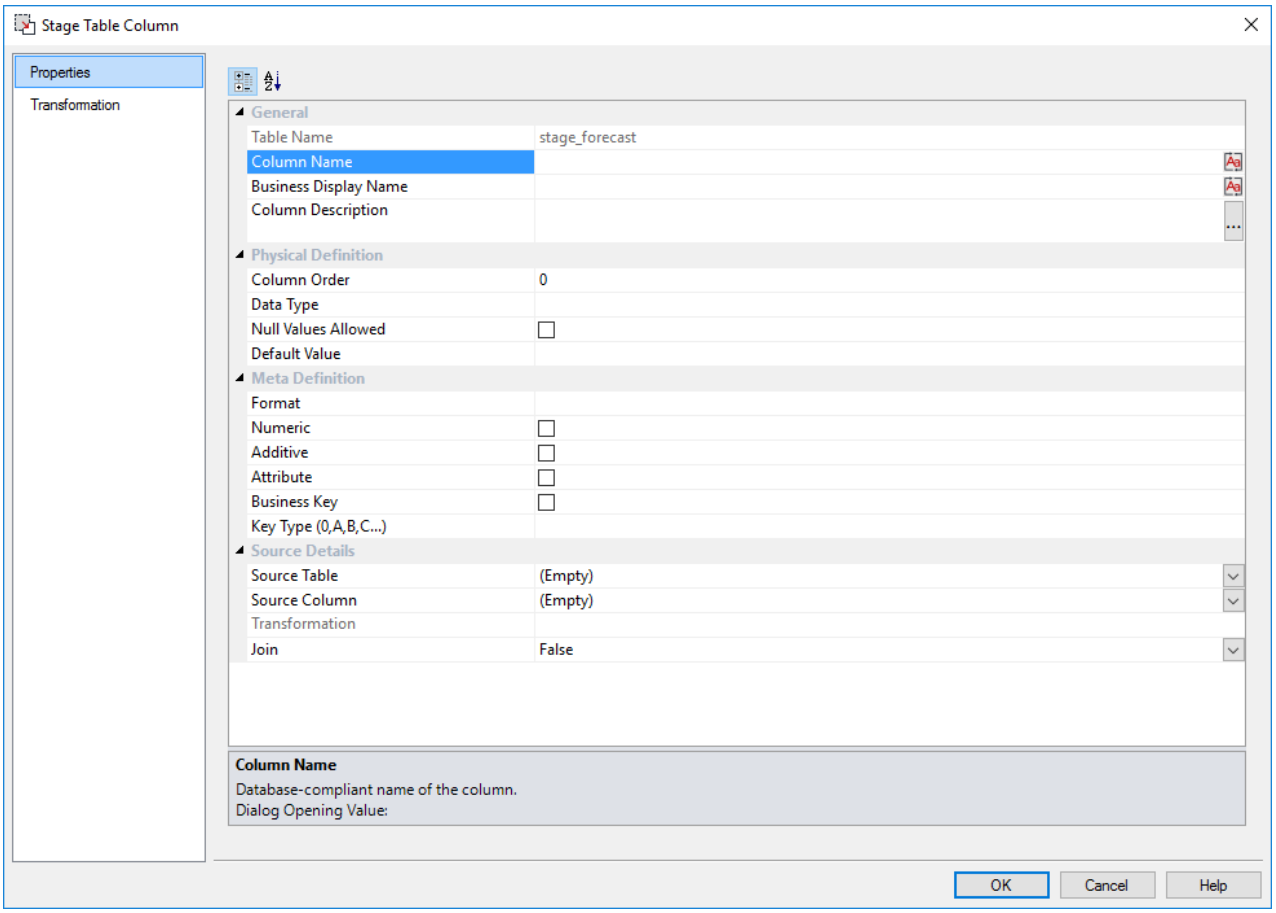
- Selecting the **Transformation** check box enables you to define a transformation for the selected column(s). Refer to **Column Transformations** for details.

Add Column

To add a column, right-click on one of the columns in the middle pane and select **Add Column**.



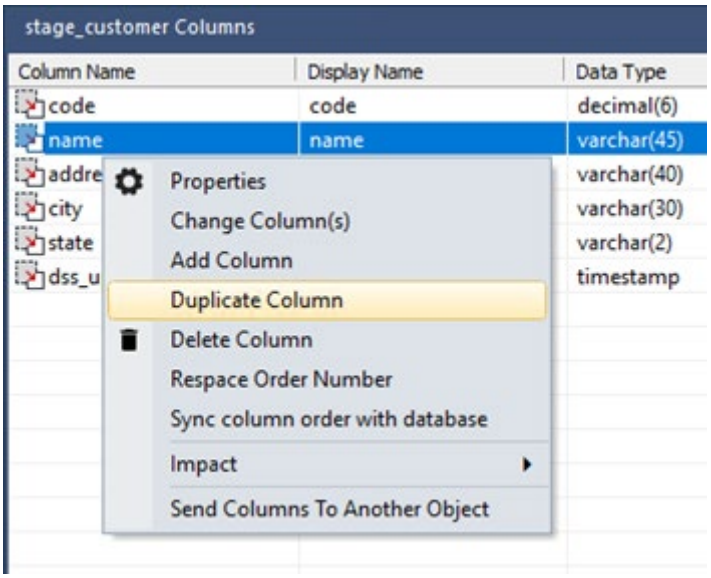
Enter the details to define a new column and click **OK**.



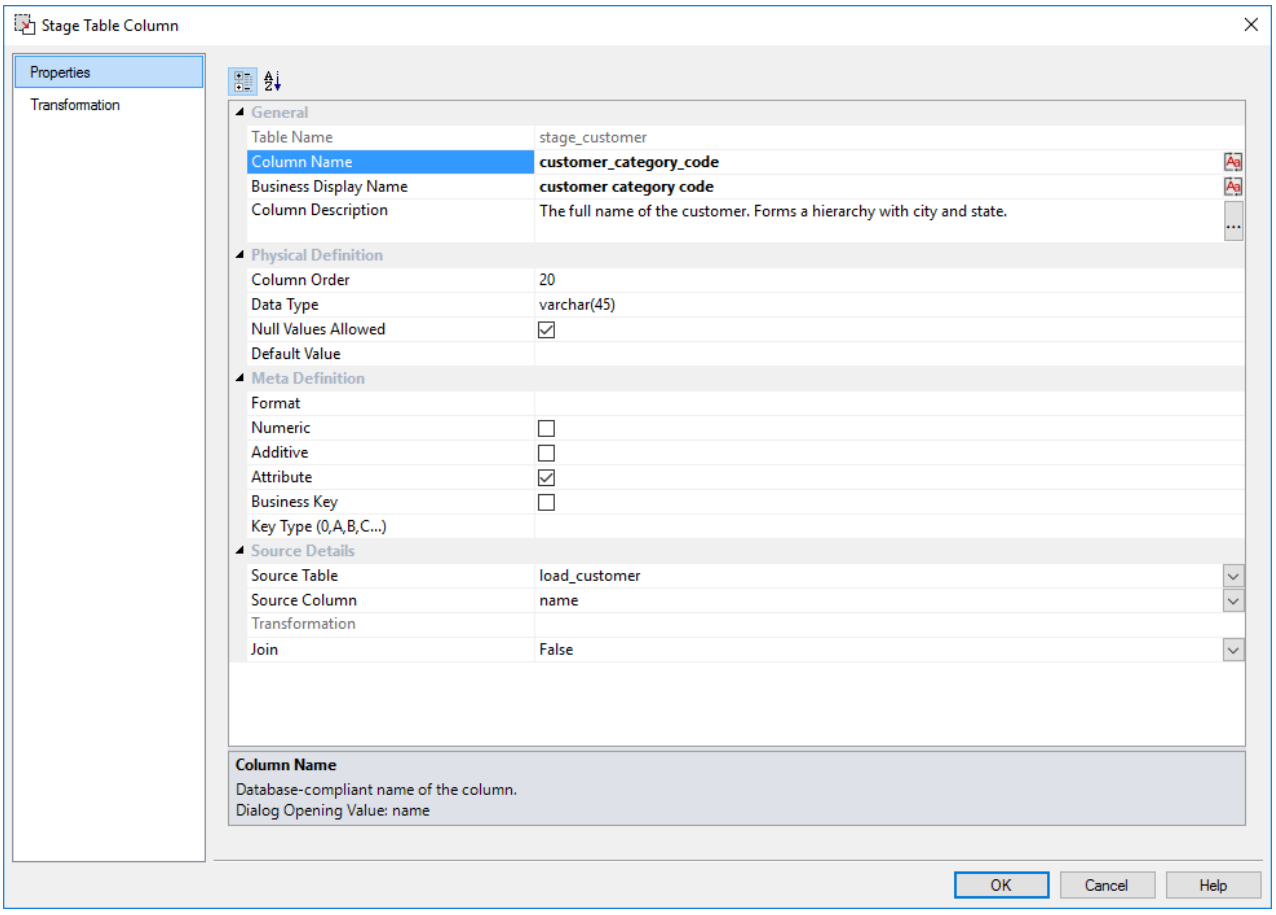
© Copyright, WhereScape Software Ltd All rights reserved.

Duplicate Column

To copy a column, right-click on one of the columns in the middle pane and select **Duplicate Column**.



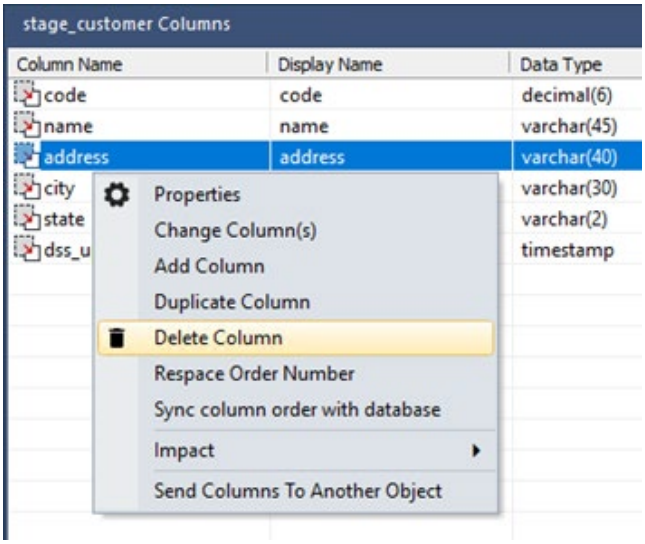
Change the **Column Name** and the **Business Display Name** and any other properties to define a new column and click **OK**.



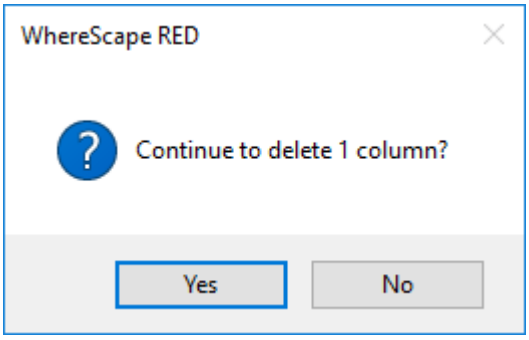
© Copyright, WhereScape Software Ltd All rights reserved.

Delete Column

To delete a column, right-click on one of the columns in the middle pane and select **Delete Column**.

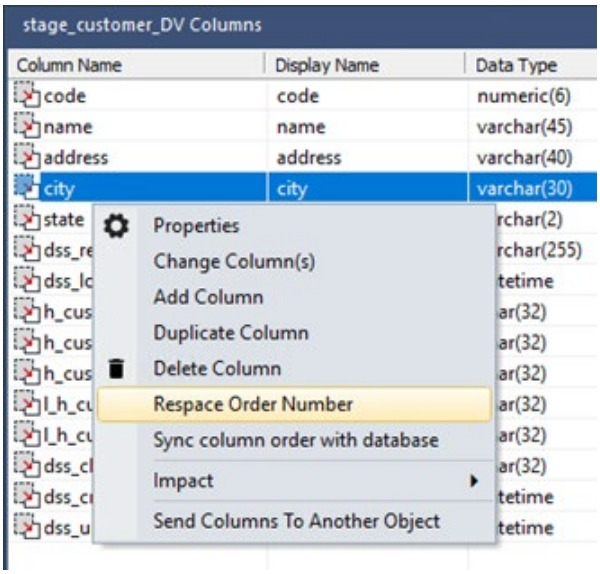


Click **Yes** to continue with the delete.



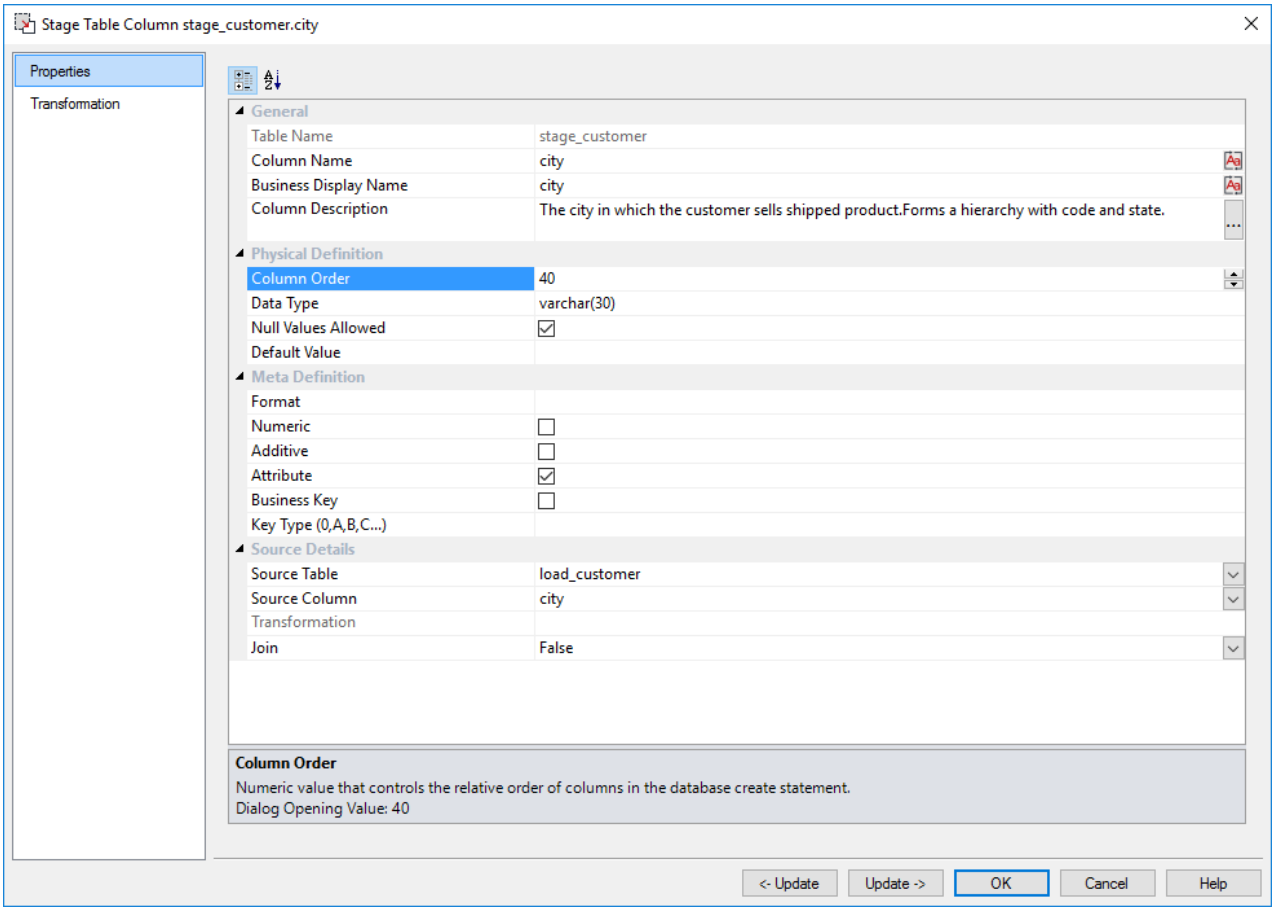
Re-space Order Number

To re-space the column order, right-click on any column in the middle pane and select **Respace Order Number.**



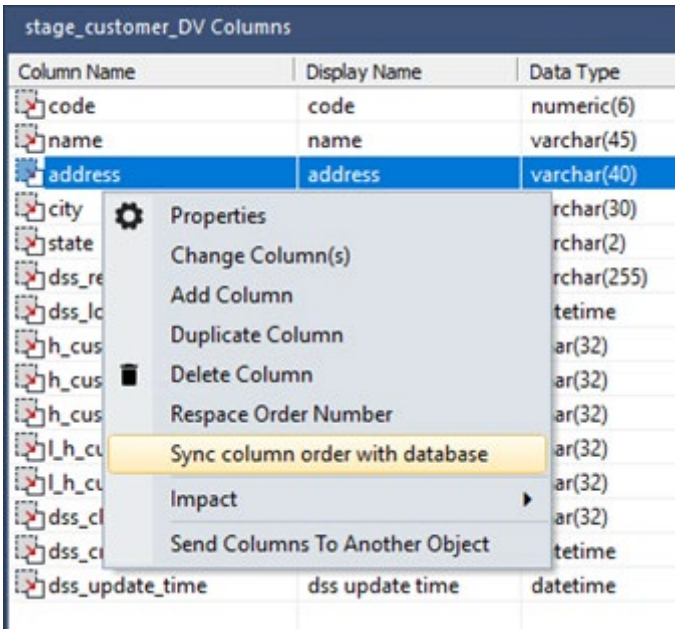
© Copyright, WhereScape Software Ltd All rights reserved.

The **Column Order** number for each column will be adjusted so that the column order numbers are evenly spaced.



Sync Column order with database

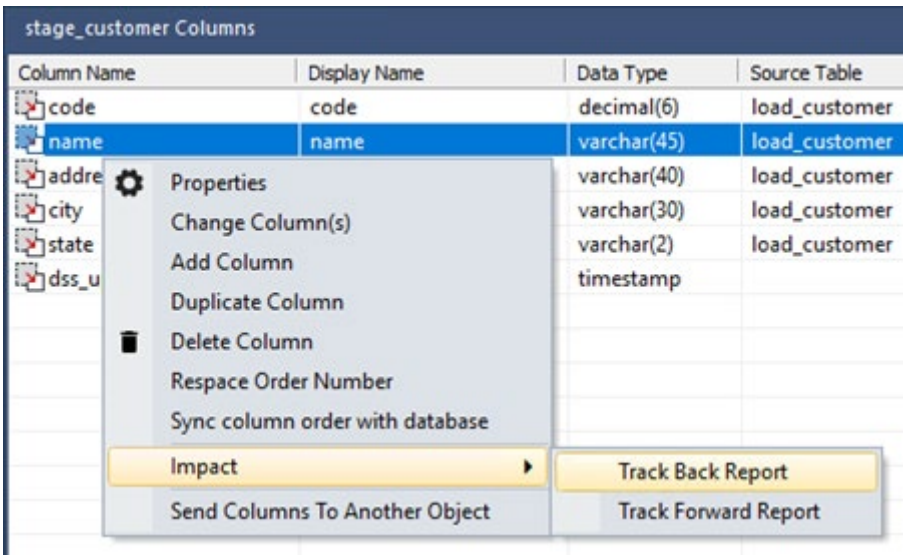
To synchronize the metadata's column order to match the same order in the physical table in the database: Right-click on one of the columns in the middle pane and select **Sync the column order with the database**. This will reorder the metadata columns to match the column order in the database table.



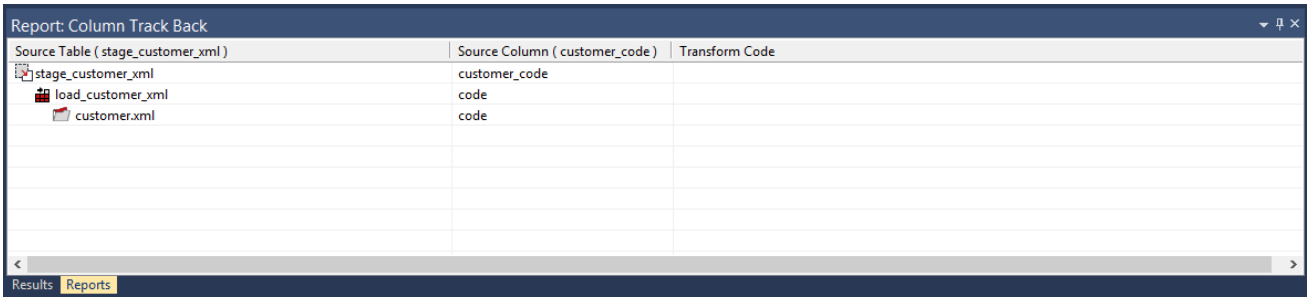
© Copyright, WhereScape Software Ltd All rights reserved.

Impact

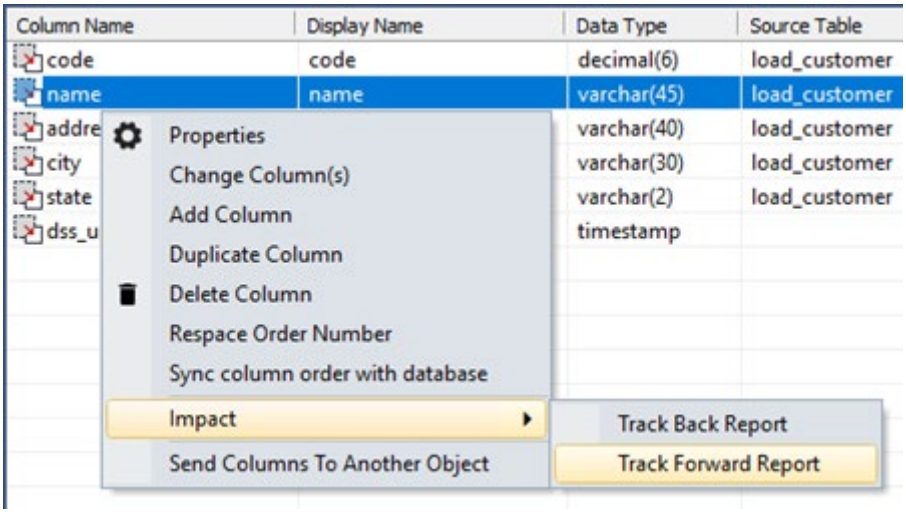
To display a Track Back Report, right-click on a column in the middle pane and select **Impact > Track Back Report**.



The report will be displayed in the bottom middle pane. This report lists the origins of the selected column. Refer to **Track Back Report** for details.

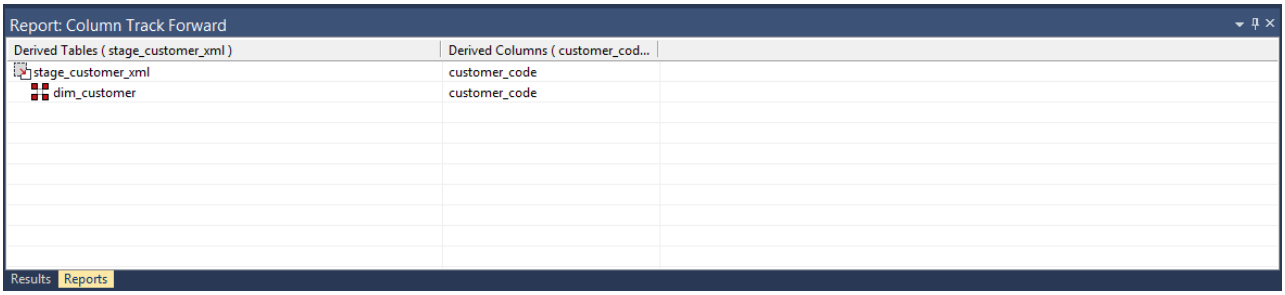


To display a Track Forward Report, right-click on a column in the middle pane and select **Impact > Track Forward Report**.



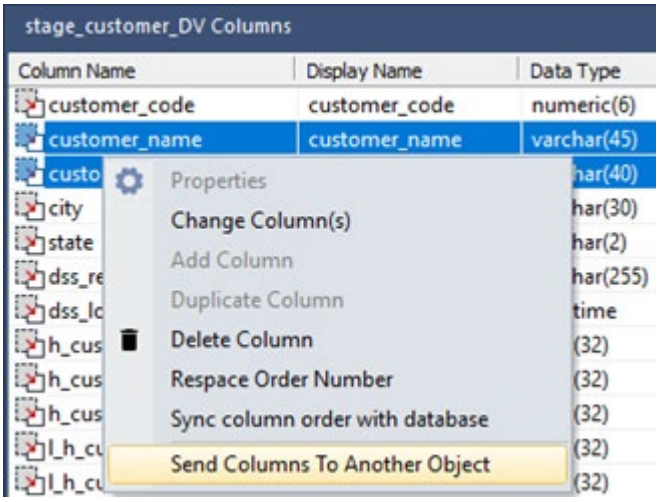
© Copyright, WhereScape Software Ltd All rights reserved.

The report will be displayed in the bottom middle pane. This report lists the columns derived from the selected column. Refer to **Track Forward Report** for details.

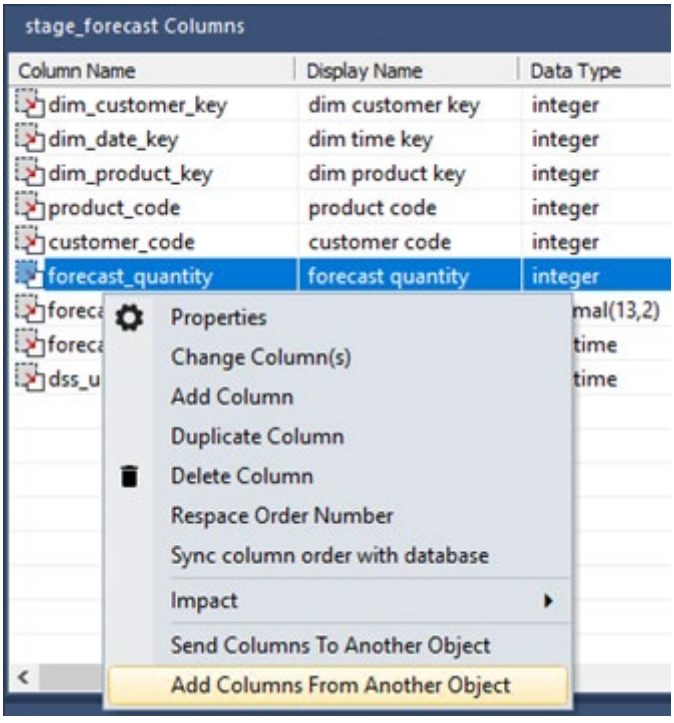


Send Columns to Another Object

To send/copy columns to another object, right-click on a column in the middle pane and select **Send Columns To Another Object**.



Click on the destination table in the left pane, then right-click in the middle pane and select **Add Columns From Another Object**.



© Copyright, WhereScape Software Ltd All rights reserved.

The columns will be added to the destination table using the same functionality and settings as drag and drop.

stage_forecast Columns

Column Name	Display Name	Data Type	Source Table	Source Column
dim_customer_key	dim customer key	integer	dim_customer	dim_customer_key
dim_date_key	dim time key	integer	dim_date	dim_date_key
dim_product_key	dim product key	integer	dim_product	dim_product_key
product_code	product code	integer	load_forecast	product_code
customer_code	customer code	integer	load_forecast	customer_code
forecast_quantity	forecast quantity	integer	load_forecast	forecast_quantity
customer_name	customer_name	varchar(45)	stage_customer	customer_name
customer_address	customer_address	varchar(40)	stage_customer	customer_address
forecast_sales_value	forecast sales value	decimal(13,2)	load_forecast	forecast_sales_value
forecast_date	forecast date	datetime	load_forecast	forecast_date
dss_update_time	dss update time	datetime		dss_update_time

© Copyright, WhereScape Software Ltd All rights reserved.

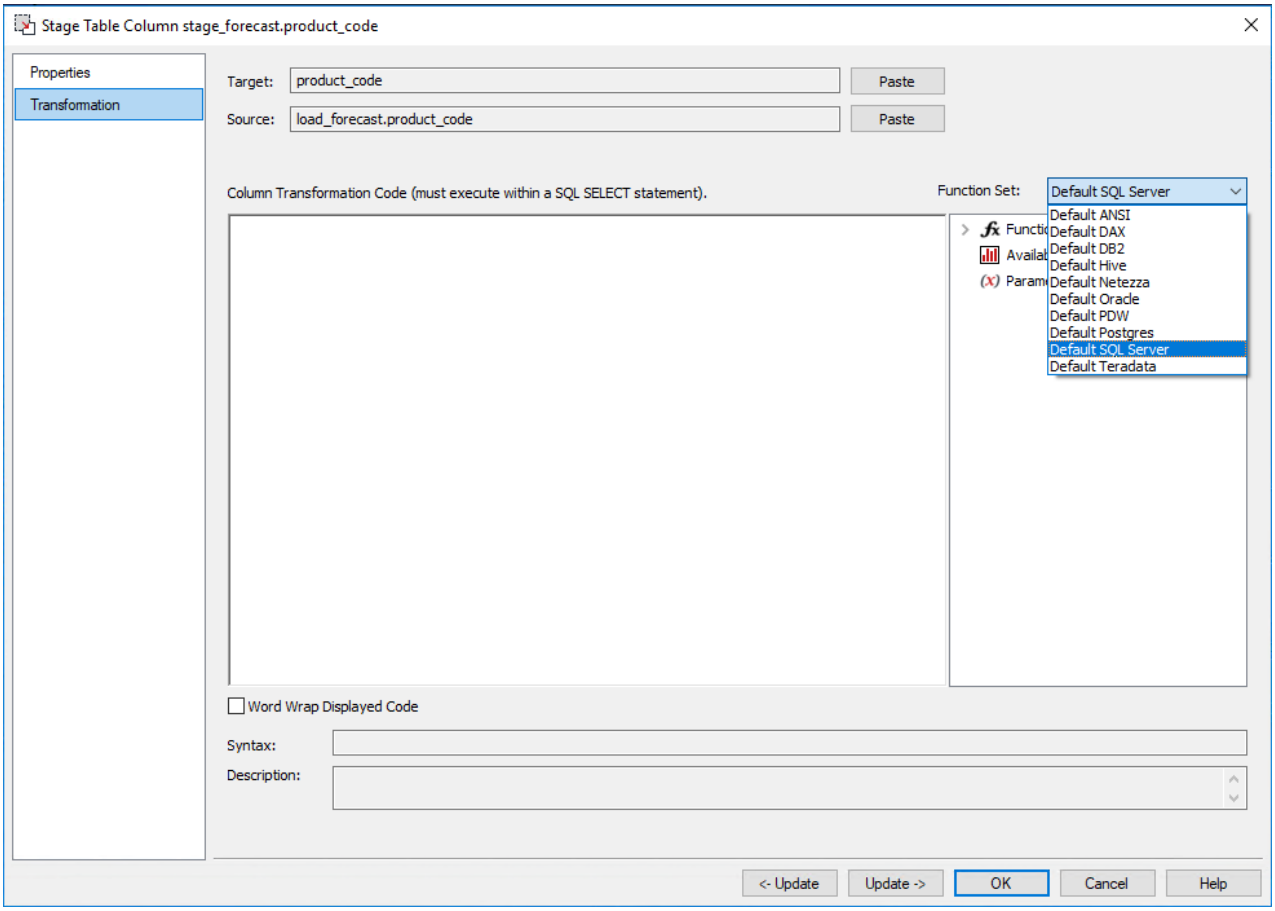
Database Functions

Using Database Function Sets

Database function sets contain a list of functions and operators that can be used for building transformations. These function sets may be created, edited, deleted, imported and exported using the **Database Functions** options from the **Tools** menu.

Column Transformation Properties

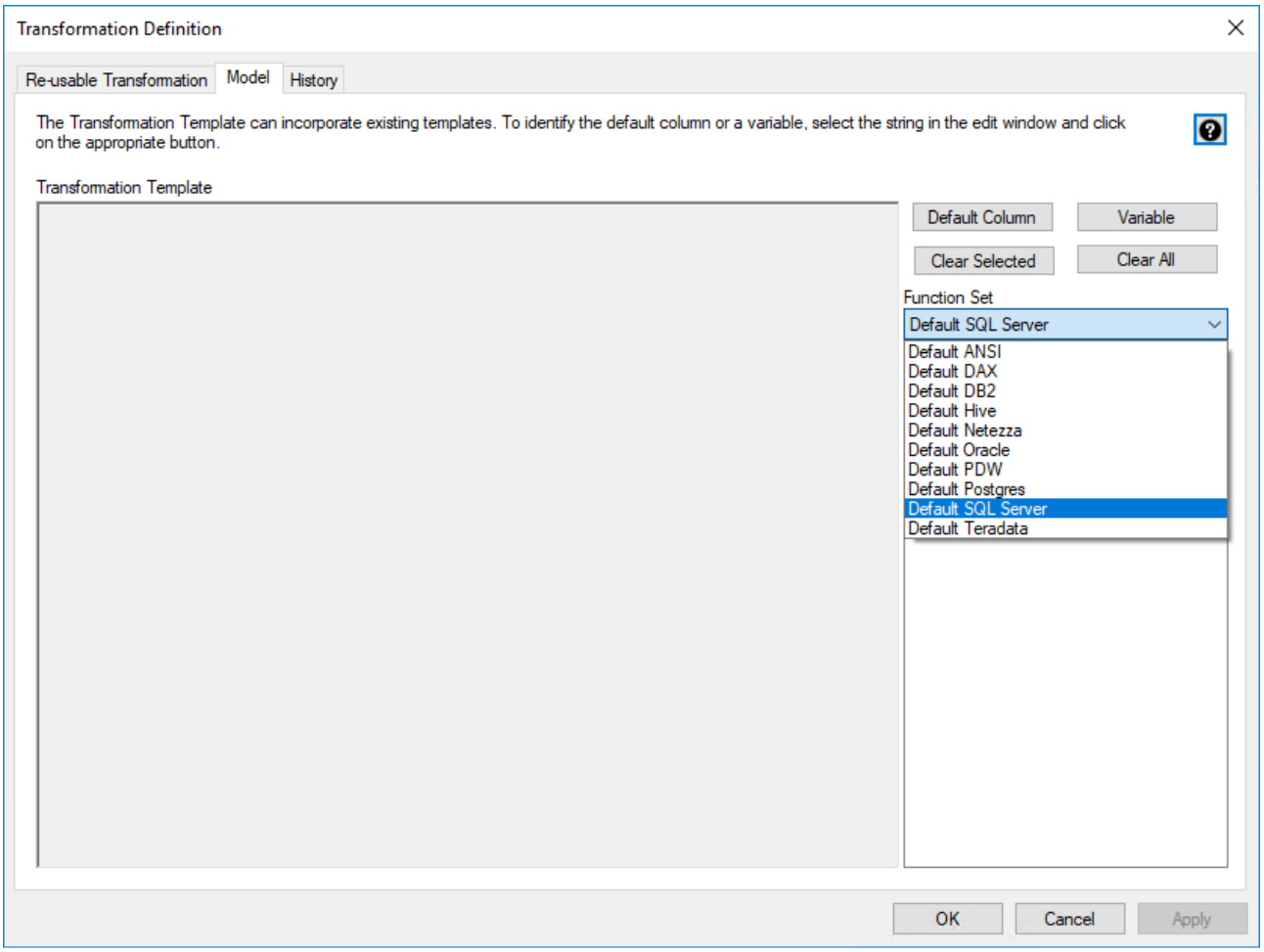
A drop-down list enables you to select which set of functions are to be displayed in the tree view when creating a transformation on a column of a table.



© Copyright, WhereScape Software Ltd All rights reserved.

Transformation Definition

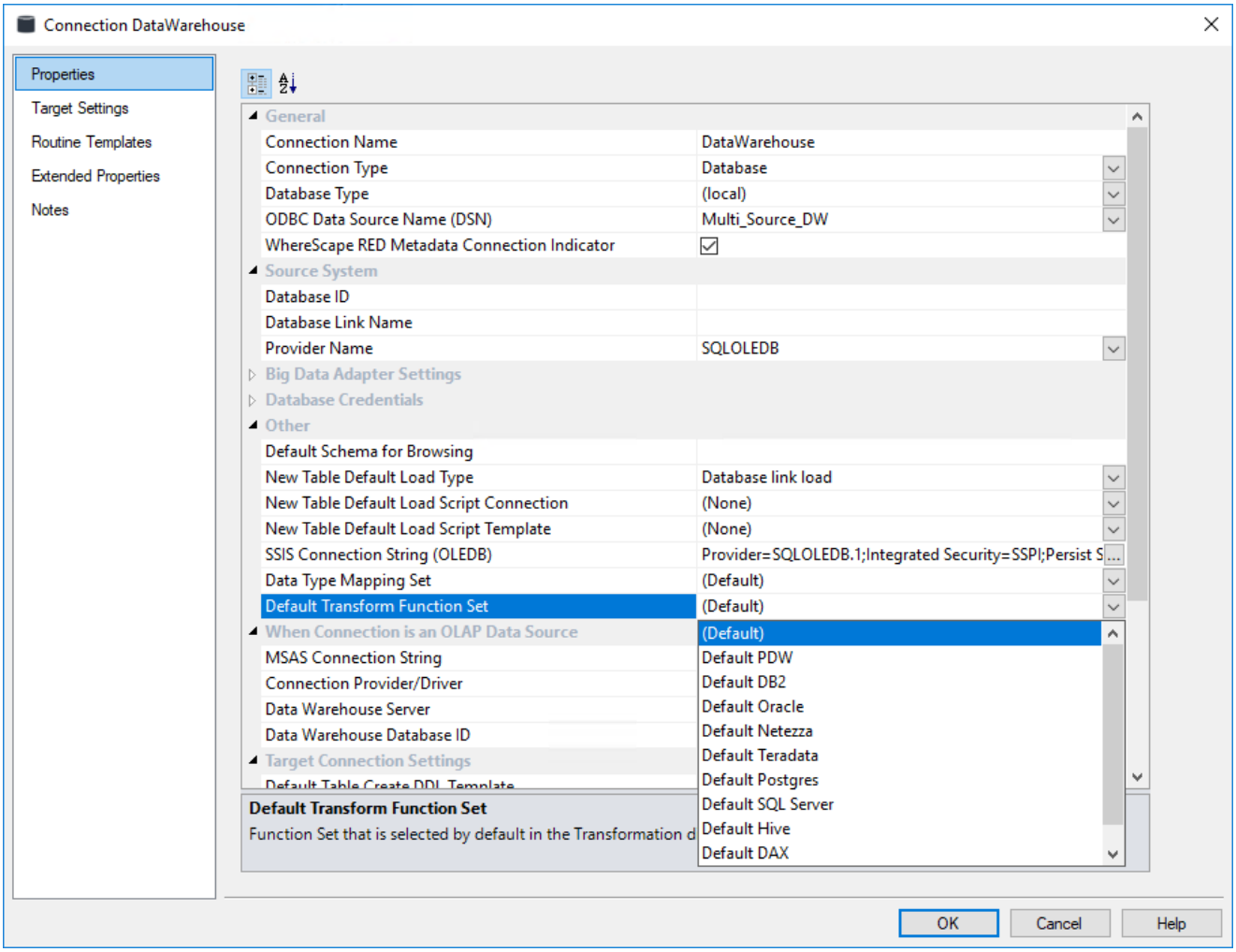
A drop-down list enables you to select which set of functions are to be displayed in the tree view when creating a re-usable transformation using **Tools > Define Re-Usable Transformations**.



© Copyright, WhereScape Software Ltd All rights reserved.

Connection Properties

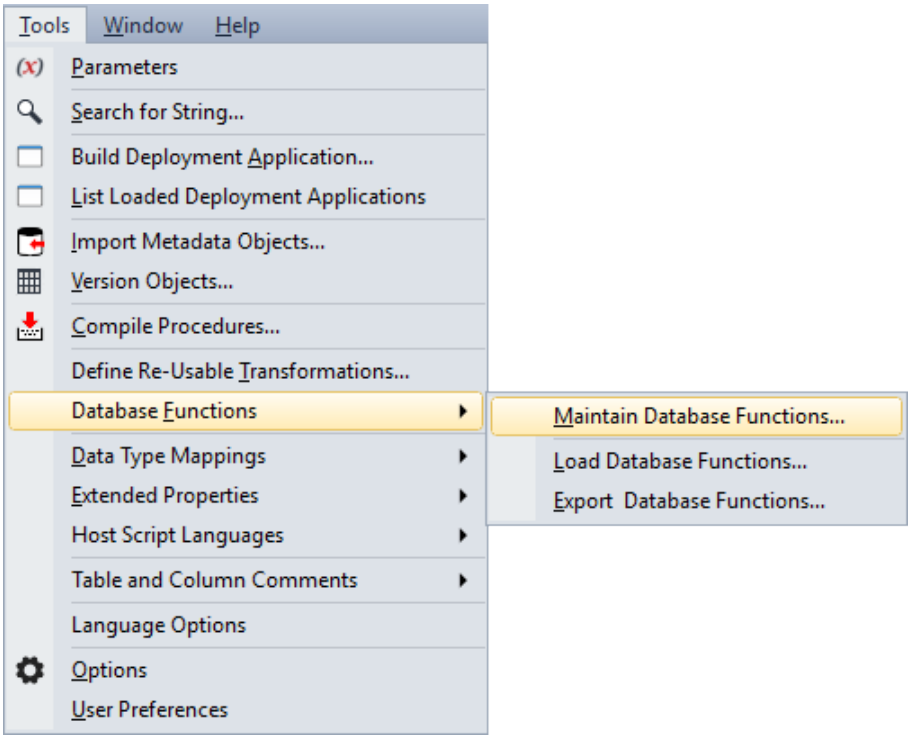
If the **Data Warehouse Metadata Connection Indicator** check-box is selected In the connection **Properties** window, a drop-down list **Default Transform Function Set** is displayed. This is the default set selected in the transformation screens above.



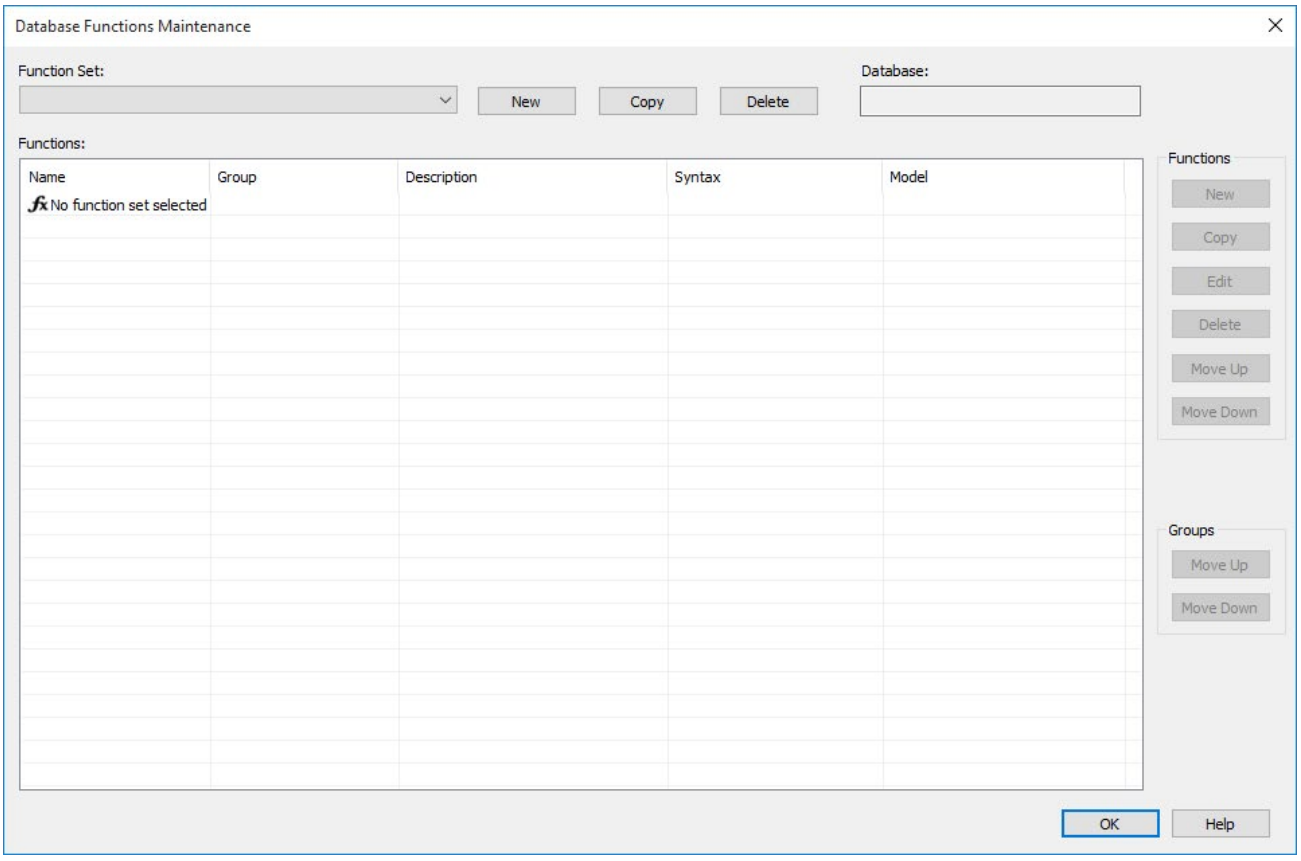
© Copyright, WhereScape Software Ltd All rights reserved.

Maintaining Database Function Sets

To maintain the database function sets, select **Tools > Database Functions > Maintain Database Functions**.

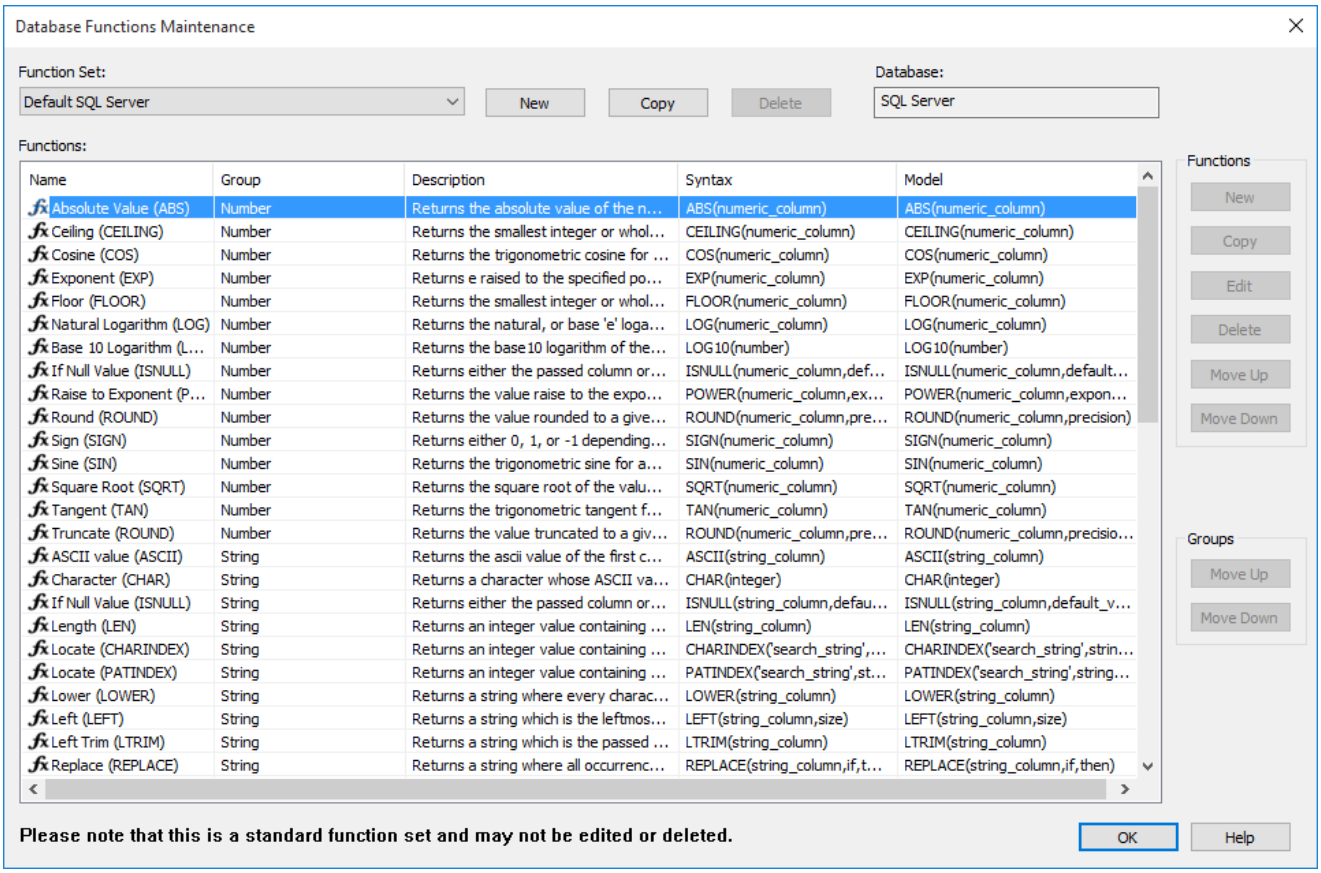


The following screen is displayed.



© Copyright, WhereScape Software Ltd All rights reserved.

Select a database function set from the **Function Set** drop-down list.

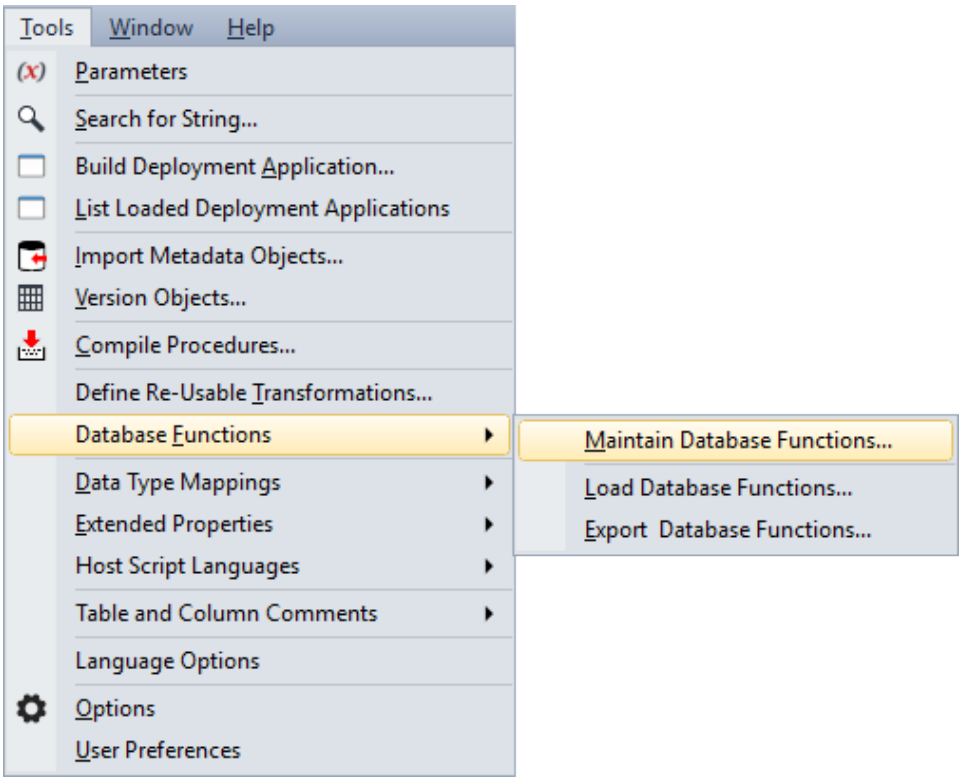


- To create a database function set, refer to **Creating a New Database Function Set**
- To copy a database function set, refer to **Copying a Database Function Set**
- To edit a database function set, refer to **Editing a Database Function Set**
- To delete a database function set, refer to **Deleting a Database Function Set**

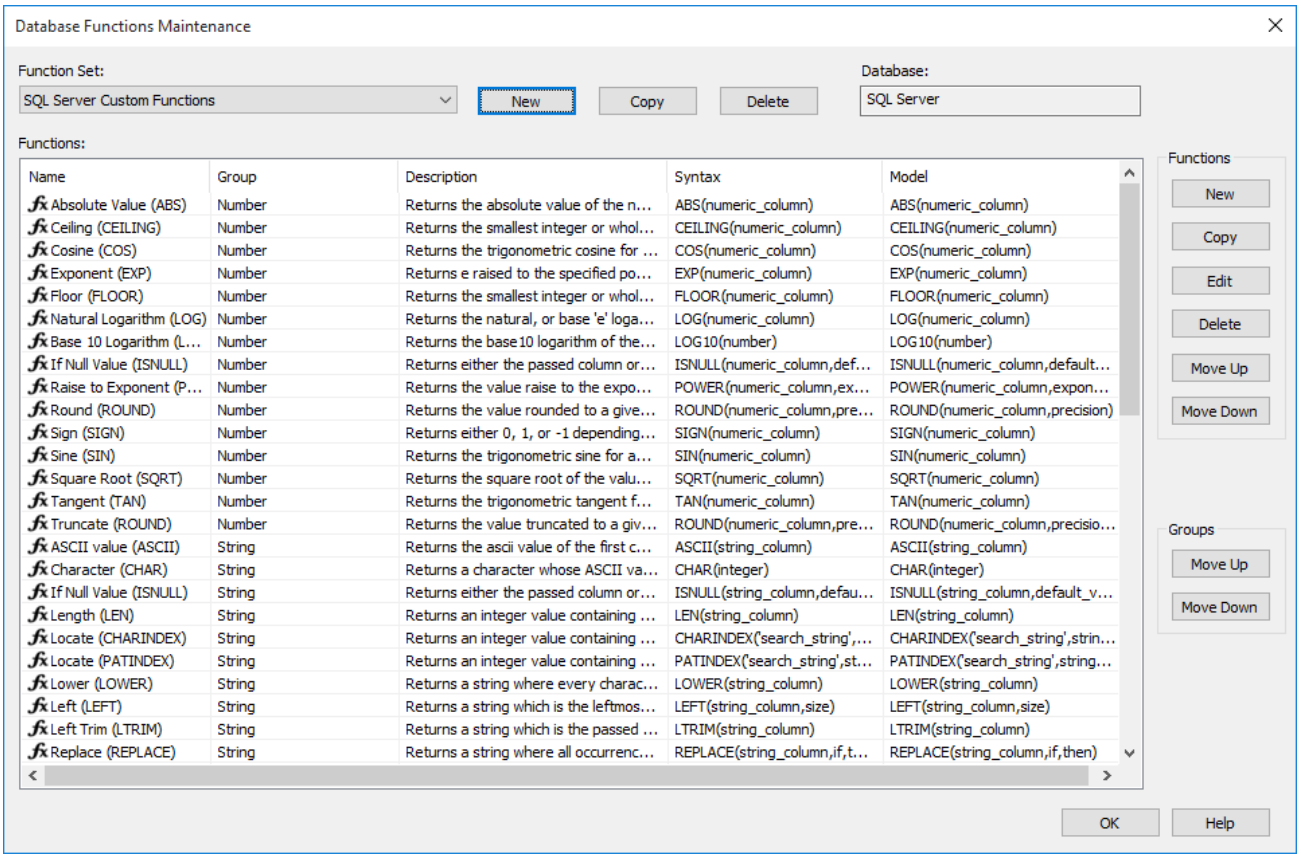
© Copyright, WhereScape Software Ltd All rights reserved.

Creating a New Database Function Set

To create a new database function set, select **Tools > Database Functions > Maintain Database Functions**.

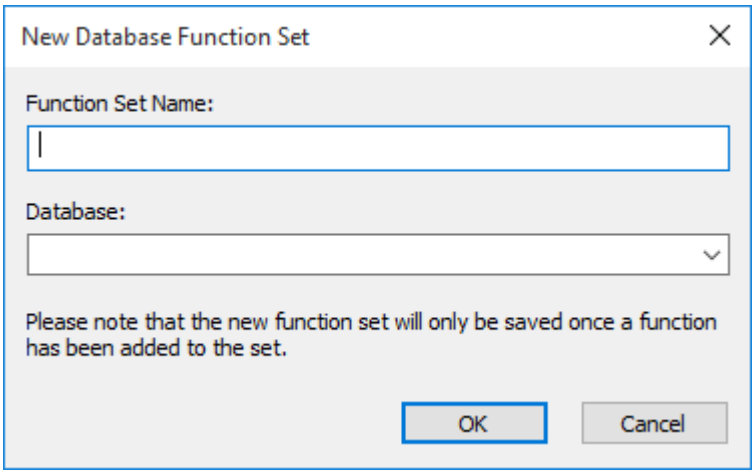


Click the **New** button.



© Copyright, WhereScape Software Ltd All rights reserved.

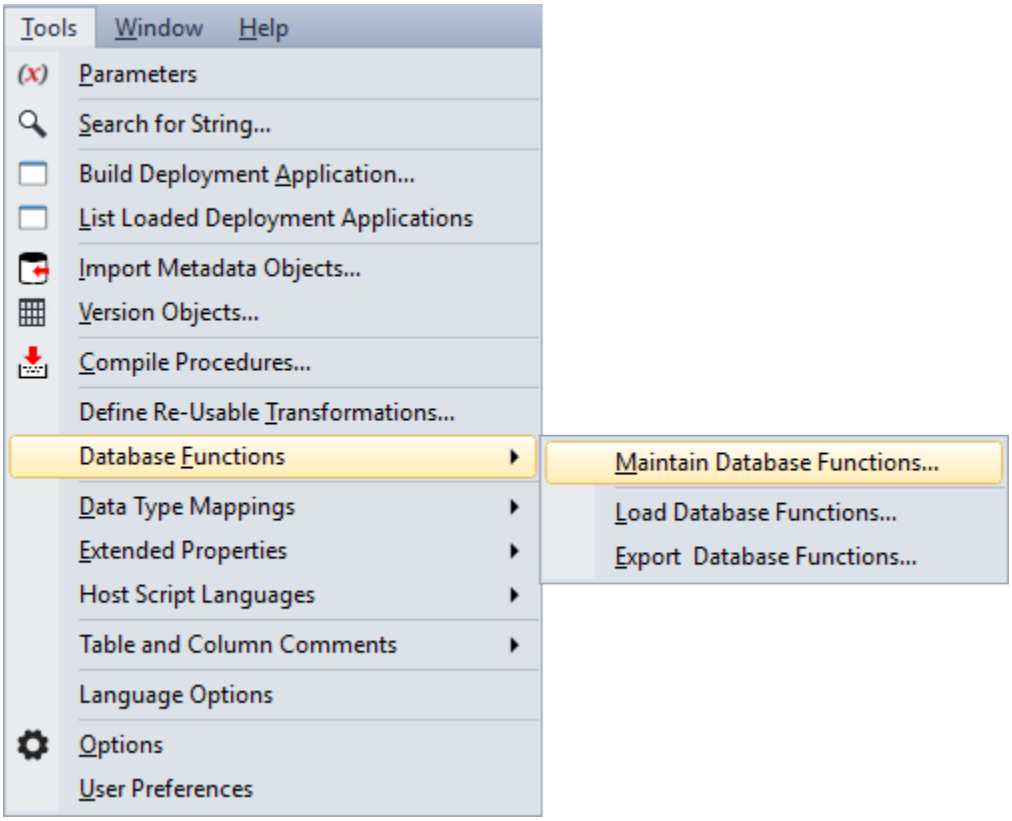
Enter a **Function Set Name** and select a **Database** from the drop-down list and then click **OK**.



Field	Description
Function Set Name	Enter a unique function set name. **Note** The function set name may not contain the phrase "Standard Functions" as this is reserved for WhereScape supplied function sets.
Database	Select the database from the drop-down list or type in the name of the database if not already in the list. This field is mandatory.

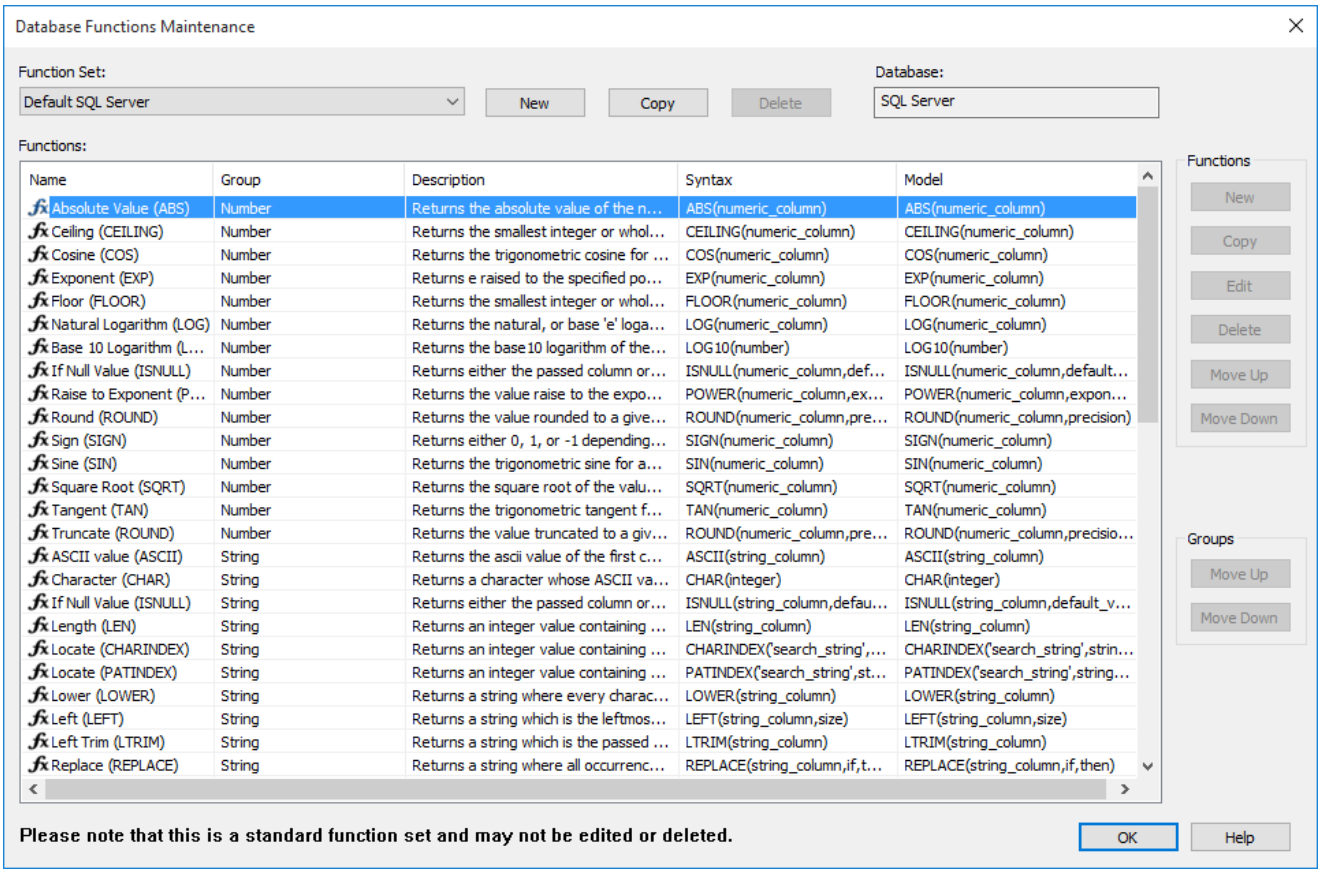
Copying a Database Function Set

To copy an existing database function set, select **Tools > Database Functions > Maintain Database Functions**.

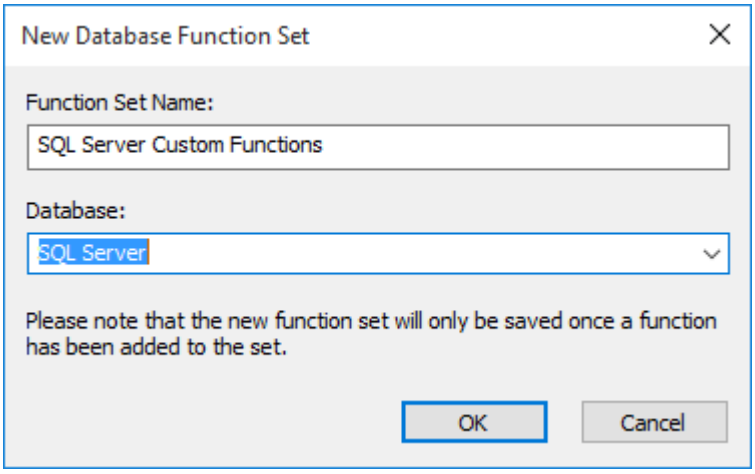


© Copyright, WhereScape Software Ltd All rights reserved.

Select a **Function Set** from the drop-down list and then click the **Copy** button.



Enter the new **Function Set Name** and select a **Database** from the drop-down list and then click **OK**.

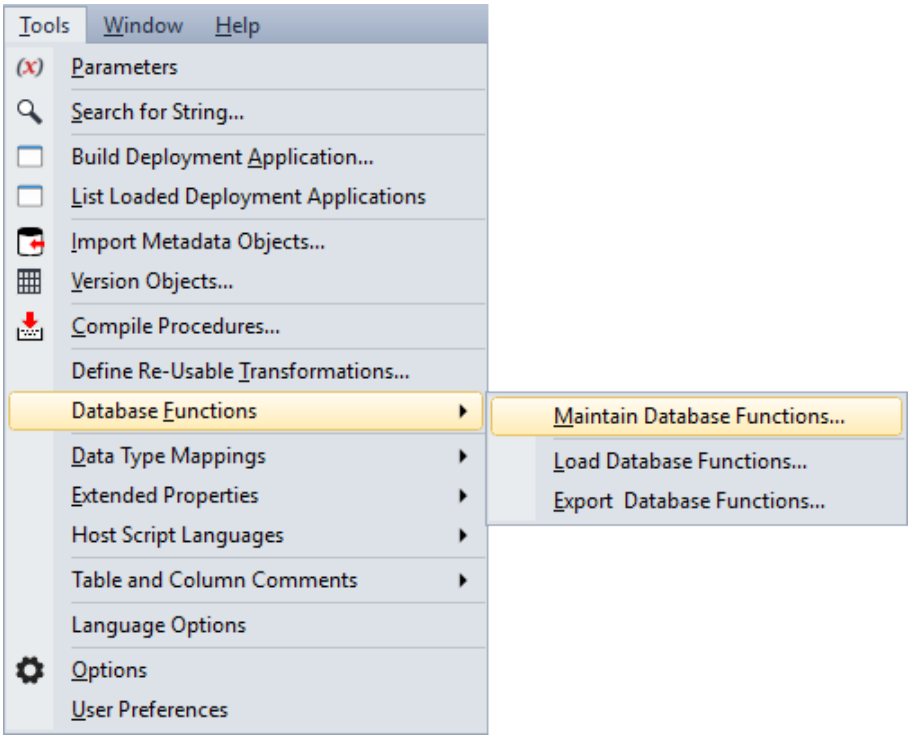


Field	Description
Function Set Name	Enter a unique function set name. **Note** The function set name may not contain the phrase "Standard Functions" as this is reserved for WhereScape supplied function sets.
Database	Select the database from the drop-down list or type in the name of the database if not already in the list. This field is mandatory.

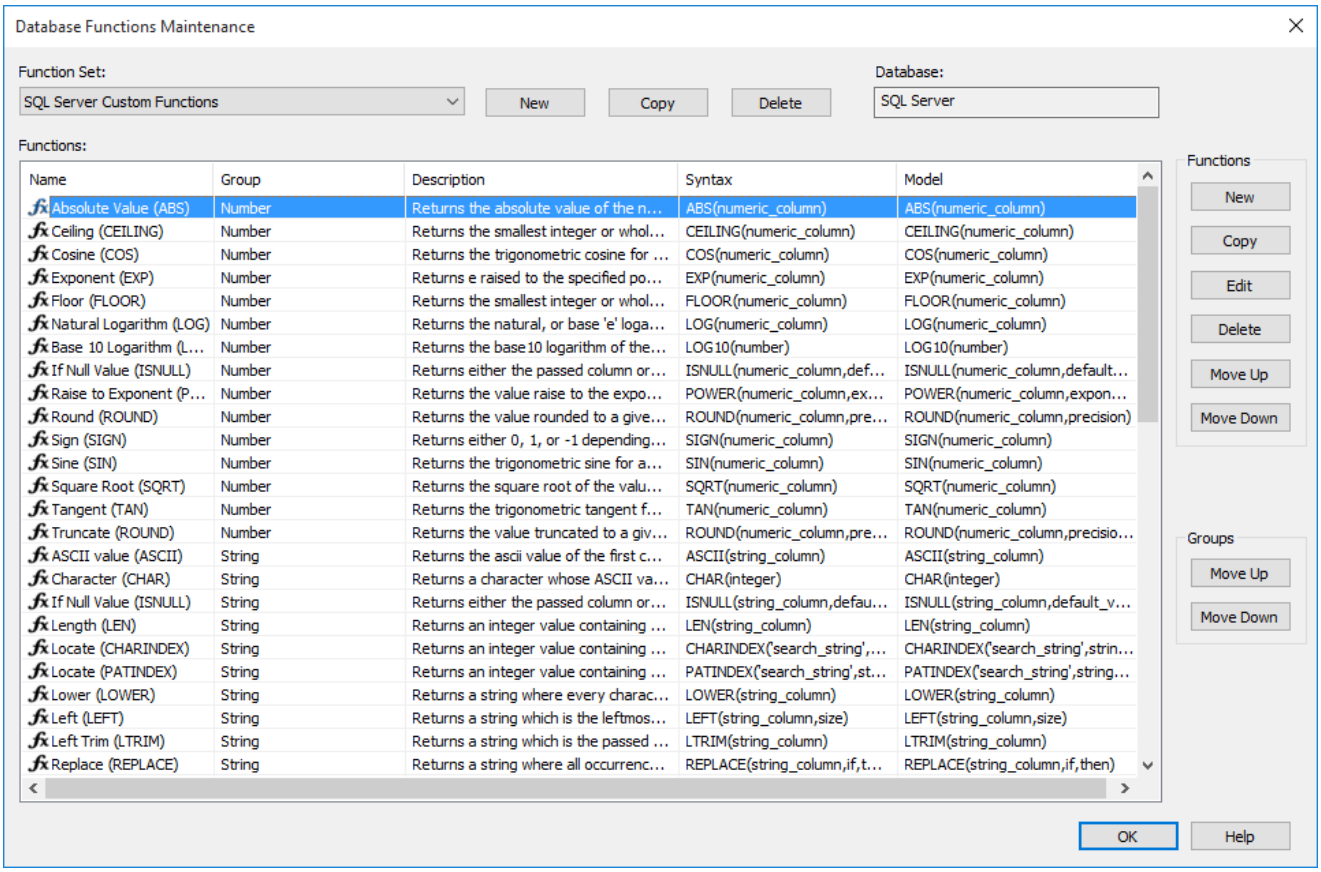
© Copyright, WhereScape Software Ltd All rights reserved.

Editing a Database Function Set

To edit a database function set, select **Tools > Database Functions > Maintain Database Functions**.



Select a database function set from the **Function Set** drop-down list.

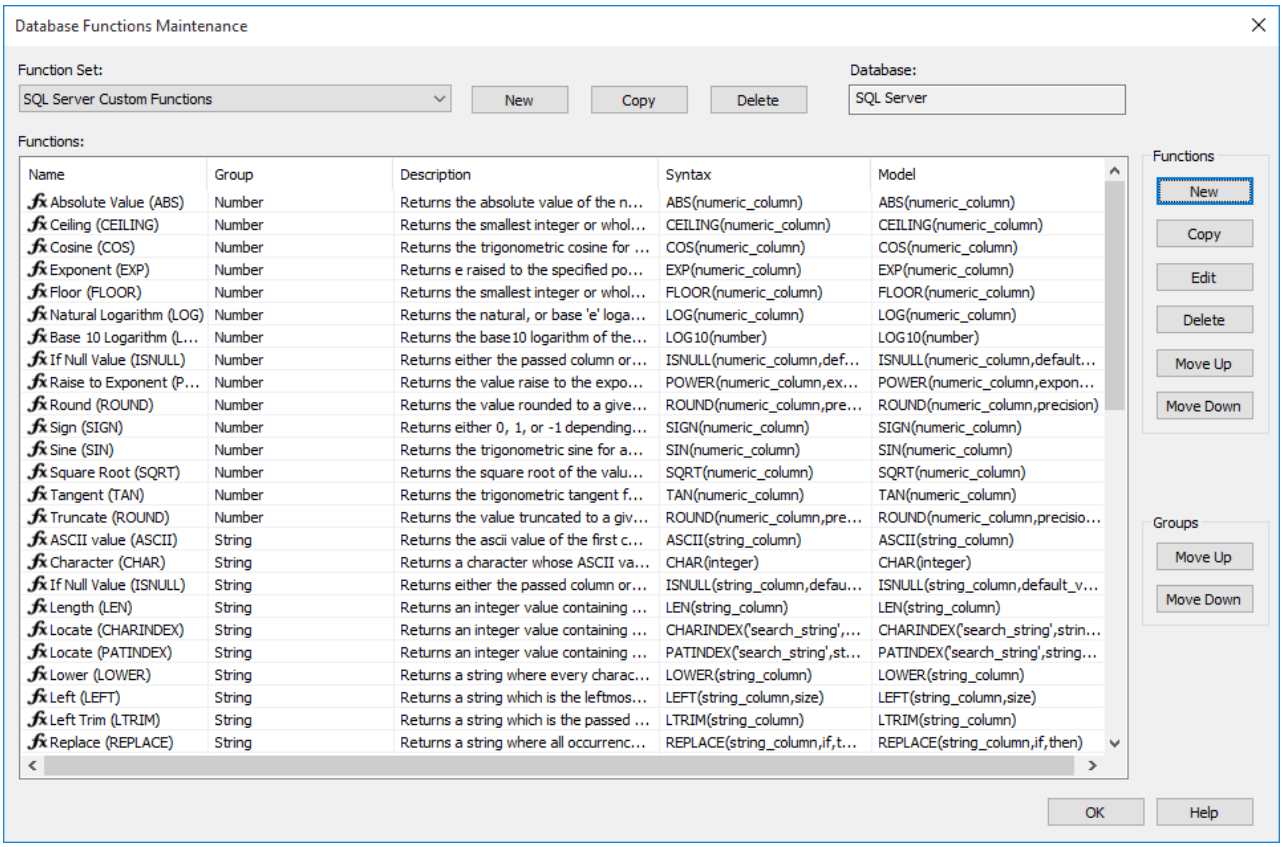


A group of buttons used to maintain the list of functions in a function set is available on the right side of the screen. These buttons are not available for standard function sets.

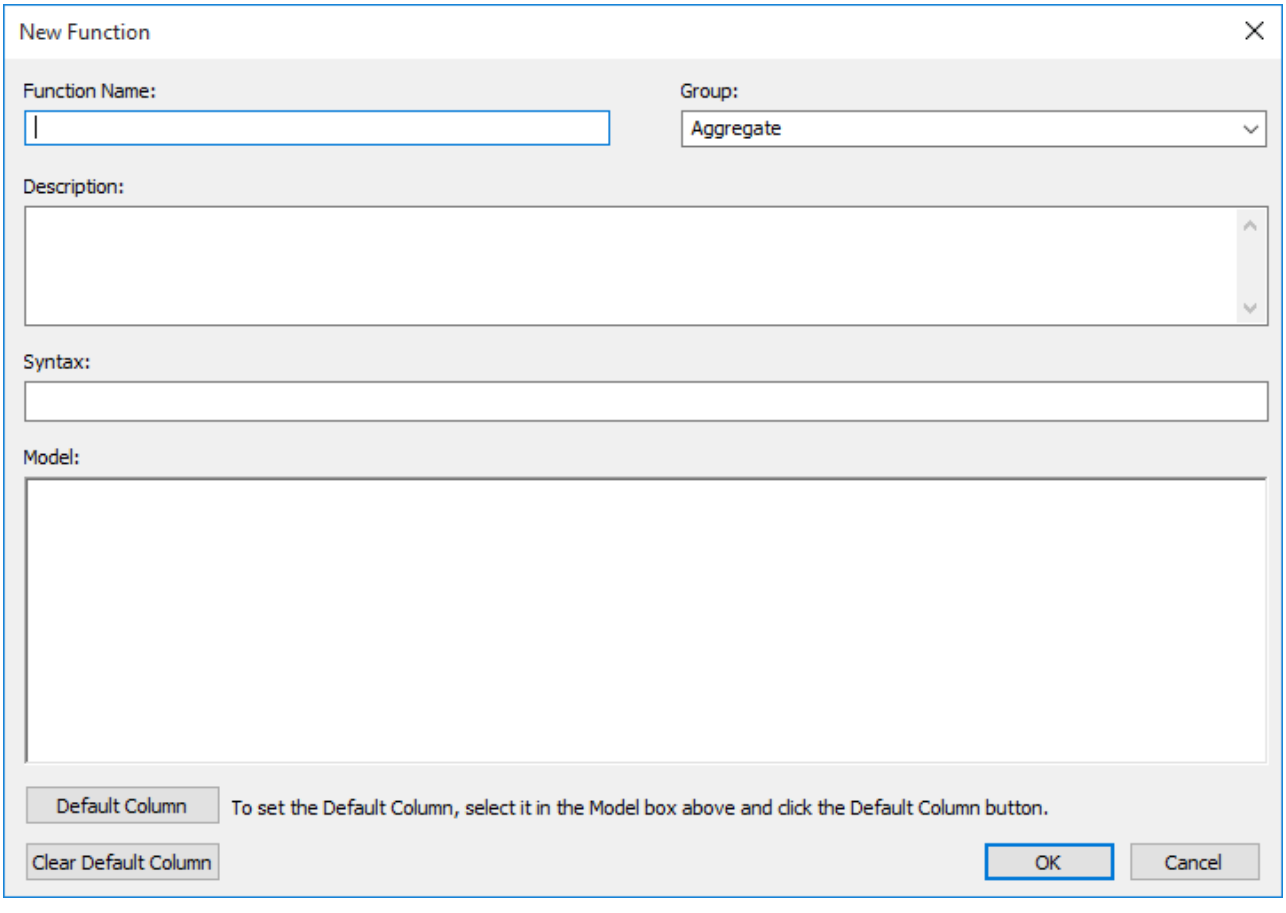
© Copyright, WhereScape Software Ltd All rights reserved.

To add a new function to the database function set:

Click the **New** button.



Enter the details for the new function and then click **OK**.



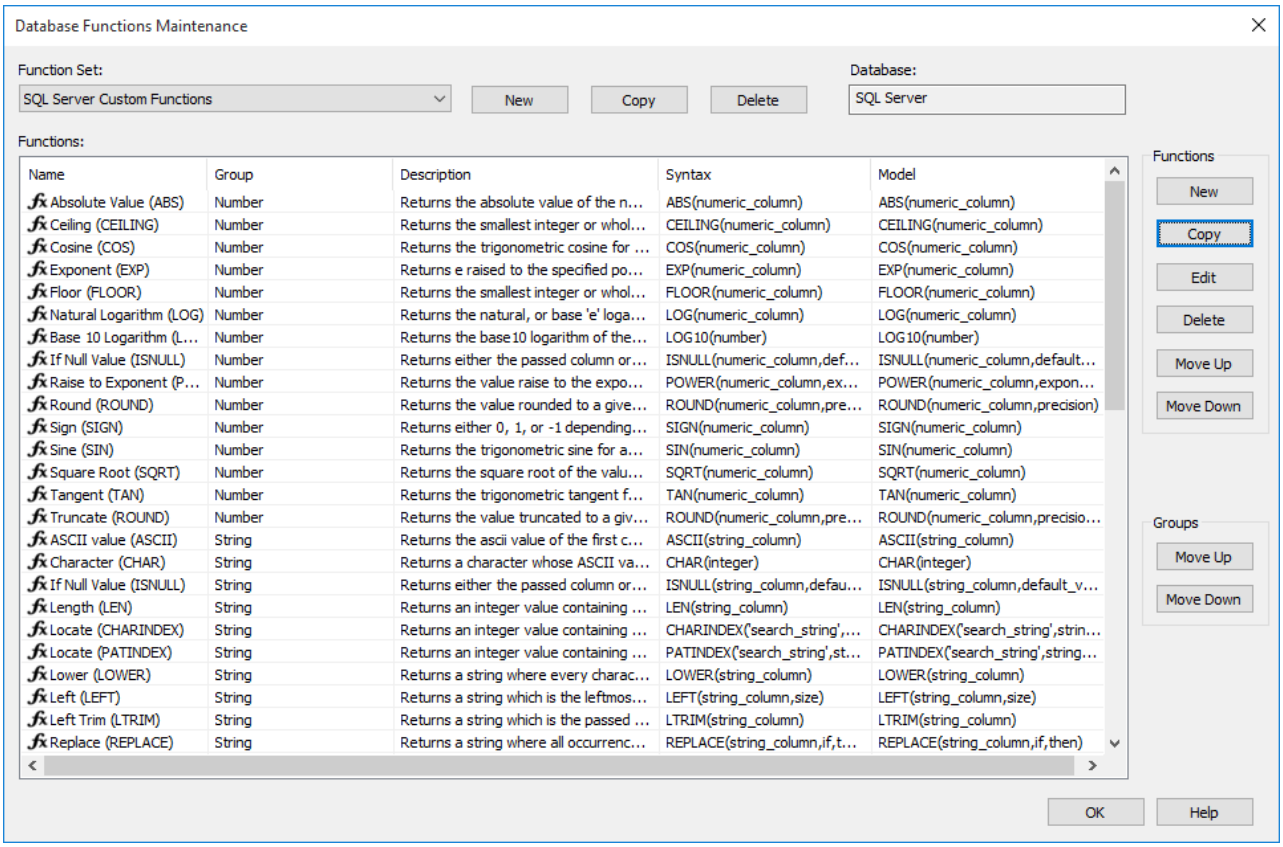
© Copyright, WhereScape Software Ltd All rights reserved.

Field	Description
Function Name	This field is mandatory and must be unique within the group.
Group	Select a group from the drop-down list or add a new group name. This field is mandatory.
Description	Enter a description for the function.
Syntax	Enter the syntax for the function.
Model	This is the text that will be pasted into the transformation/ model fields when the function is selected. This field is mandatory.
Default Column	This is the text that will automatically be highlighted when the function is used in the transformation/model screens. To set the default column, highlight it in the model field and then click the **Default Column** button. The default column is shown in red in the **Model** field.
Clear Default Column	Click this button to clear the default column.

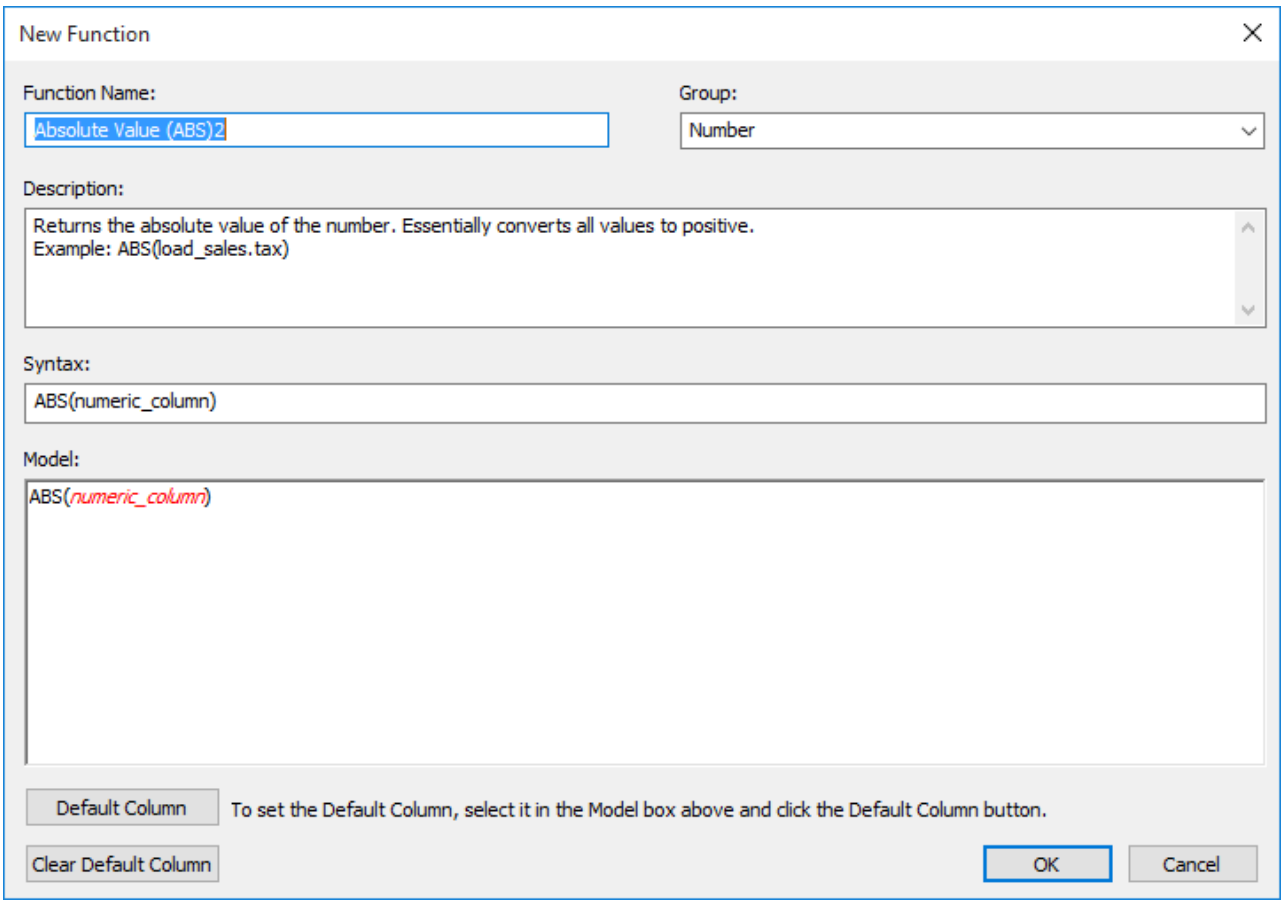
© Copyright, WhereScape Software Ltd All rights reserved.

To copy an existing function in the database function set:

Select the function and then click the **Copy** button.



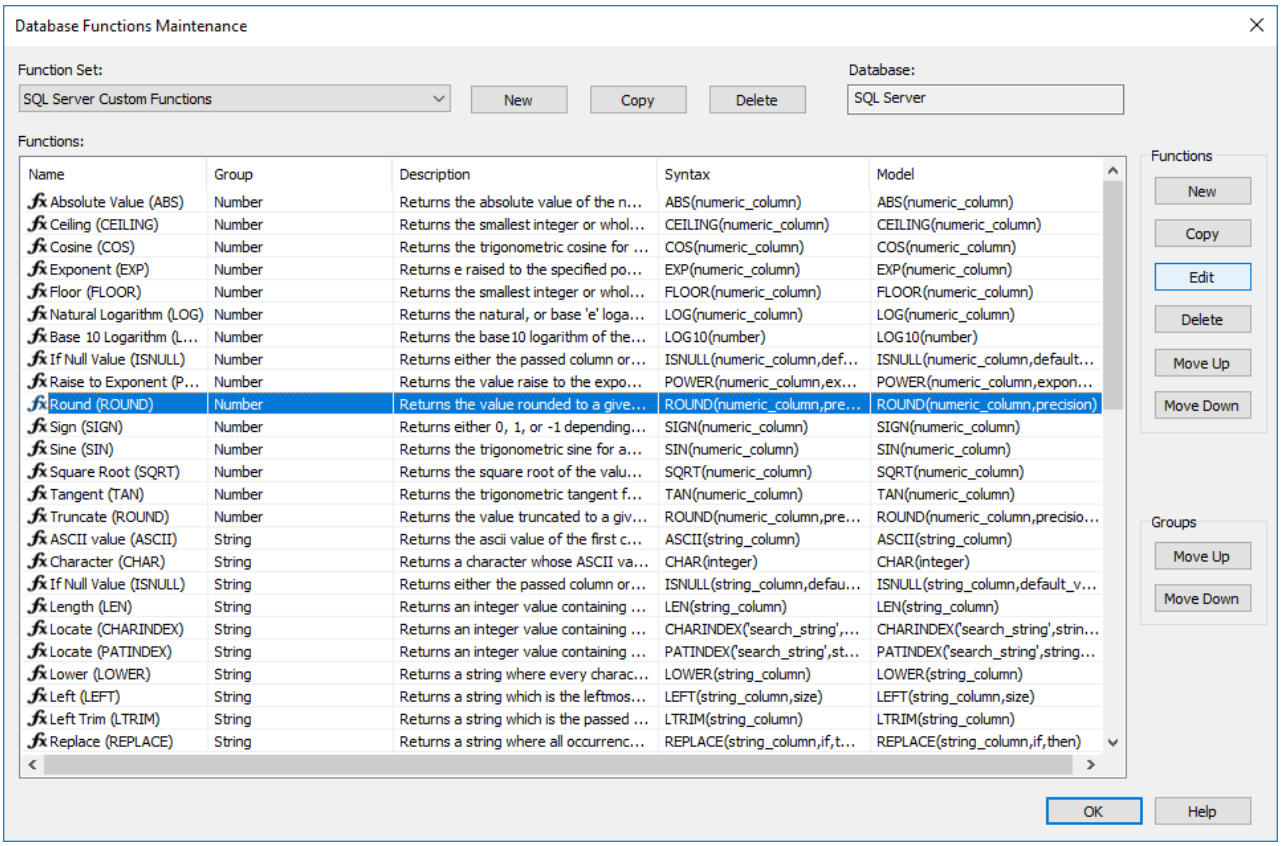
Enter a new **Function Name** and change any other details; then click **OK**.



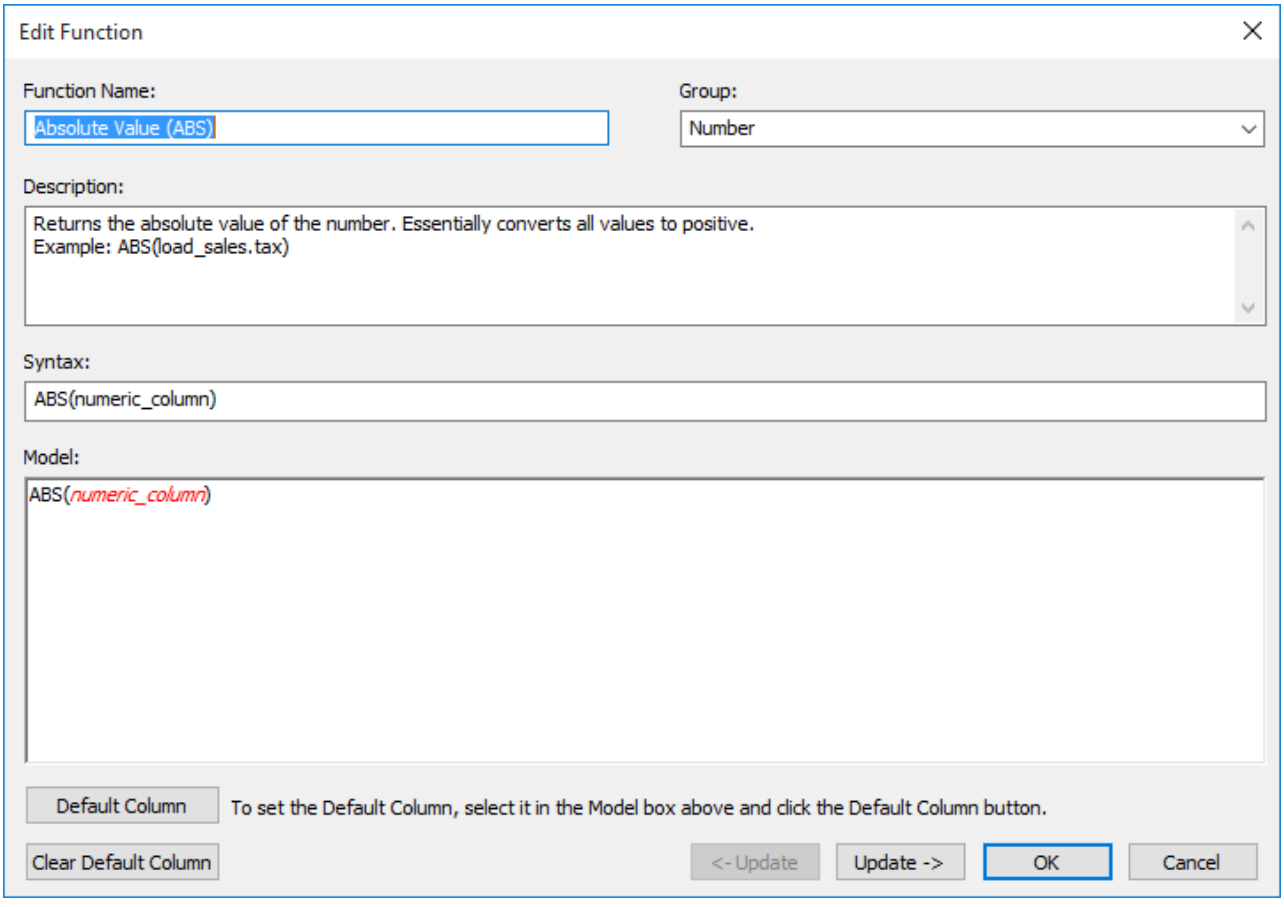
© Copyright, WhereScape Software Ltd All rights reserved.

To edit an existing function in the database function set:

Select the function and then click the **Edit** button on the right.



Change any fields as required and then click **OK**.



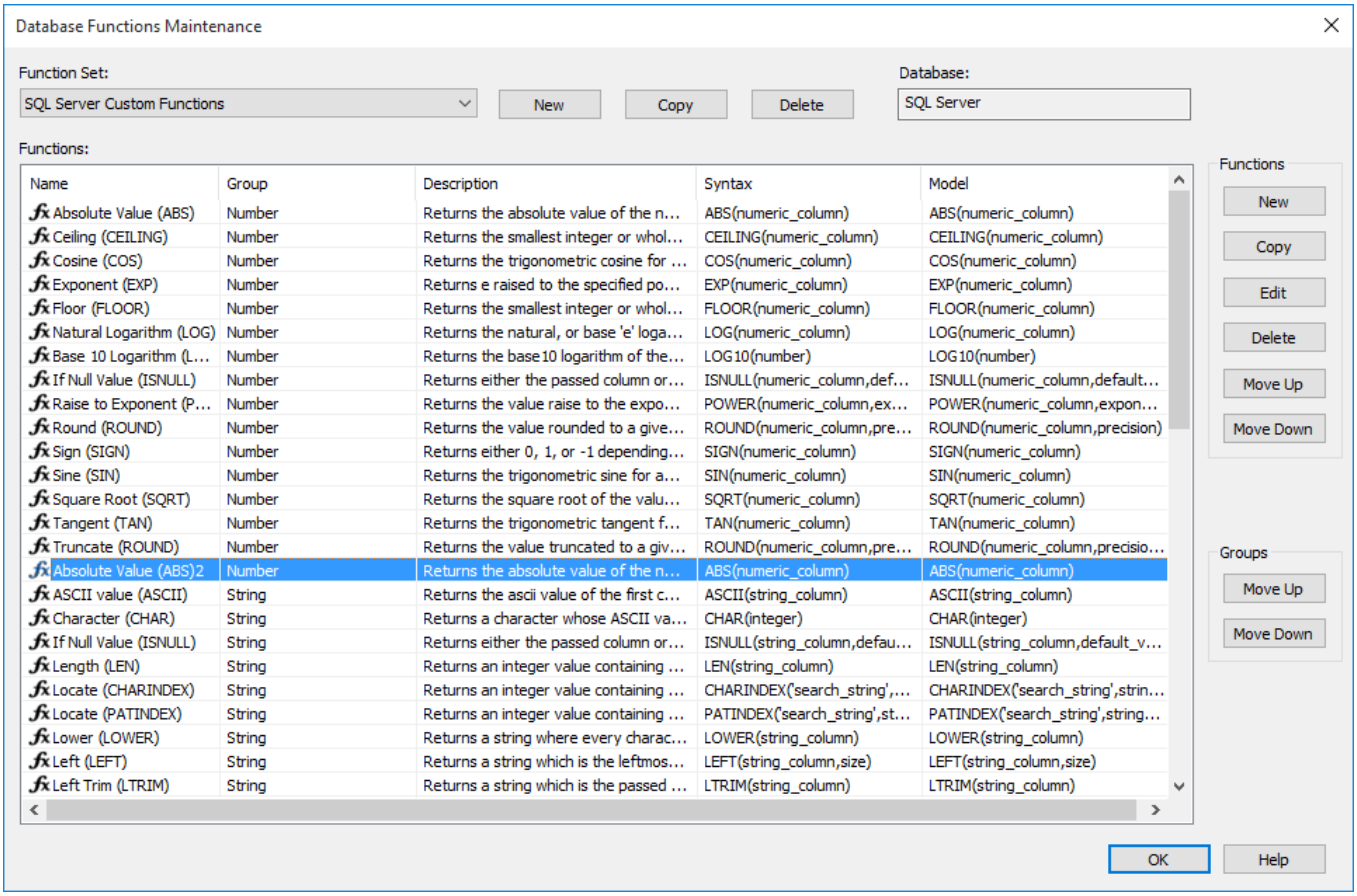
© Copyright, WhereScape Software Ltd All rights reserved.

Note

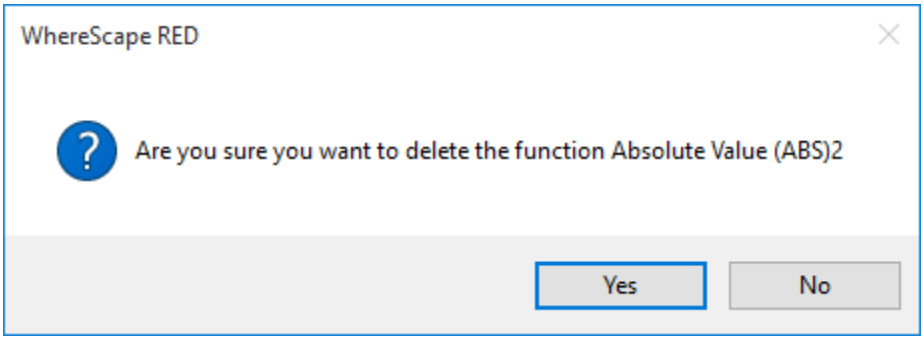
A function can also be edited by double-clicking the function.

To delete an existing function in the database function set:

Select the function and then click the **Delete** button.



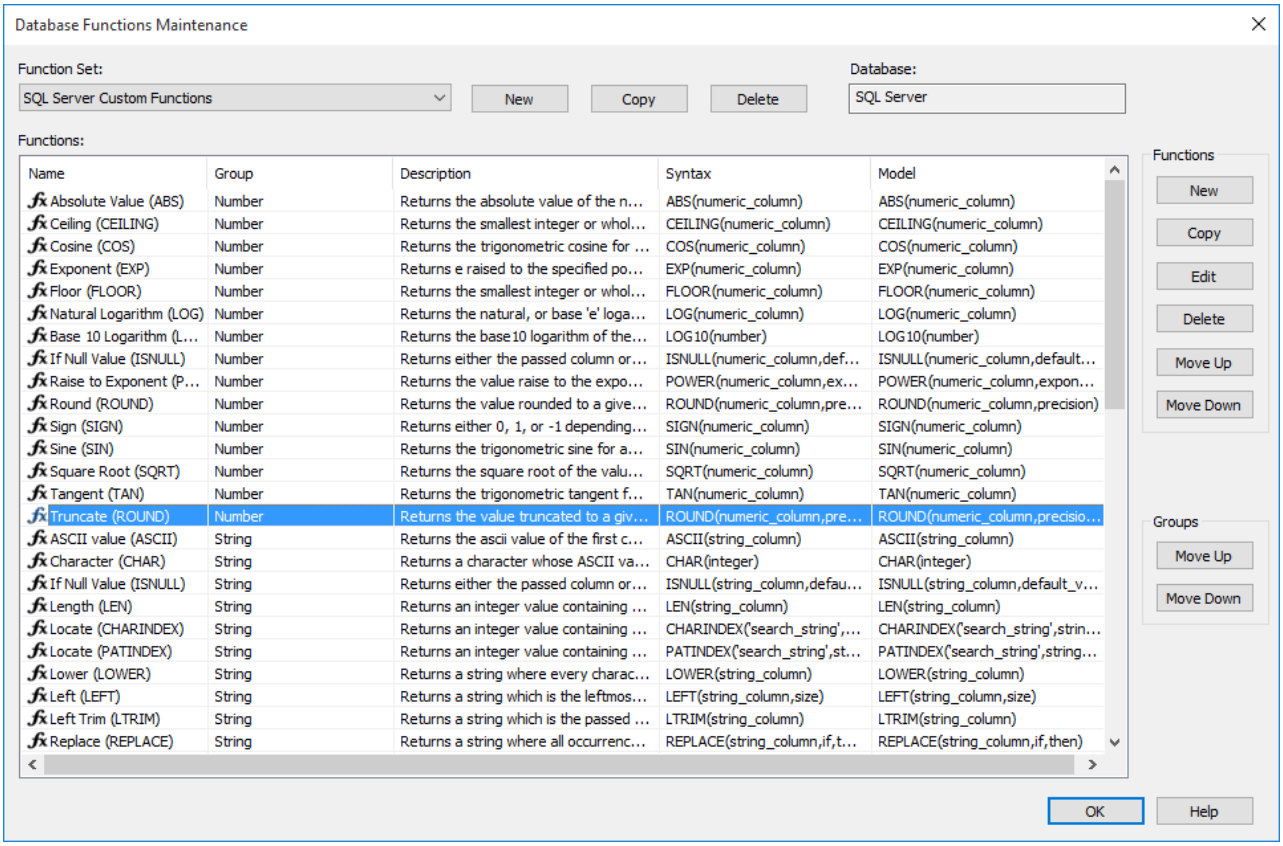
Click **Yes** to confirm the delete.



© Copyright, WhereScape Software Ltd All rights reserved.

To move a function in the database function set up or down in the list:

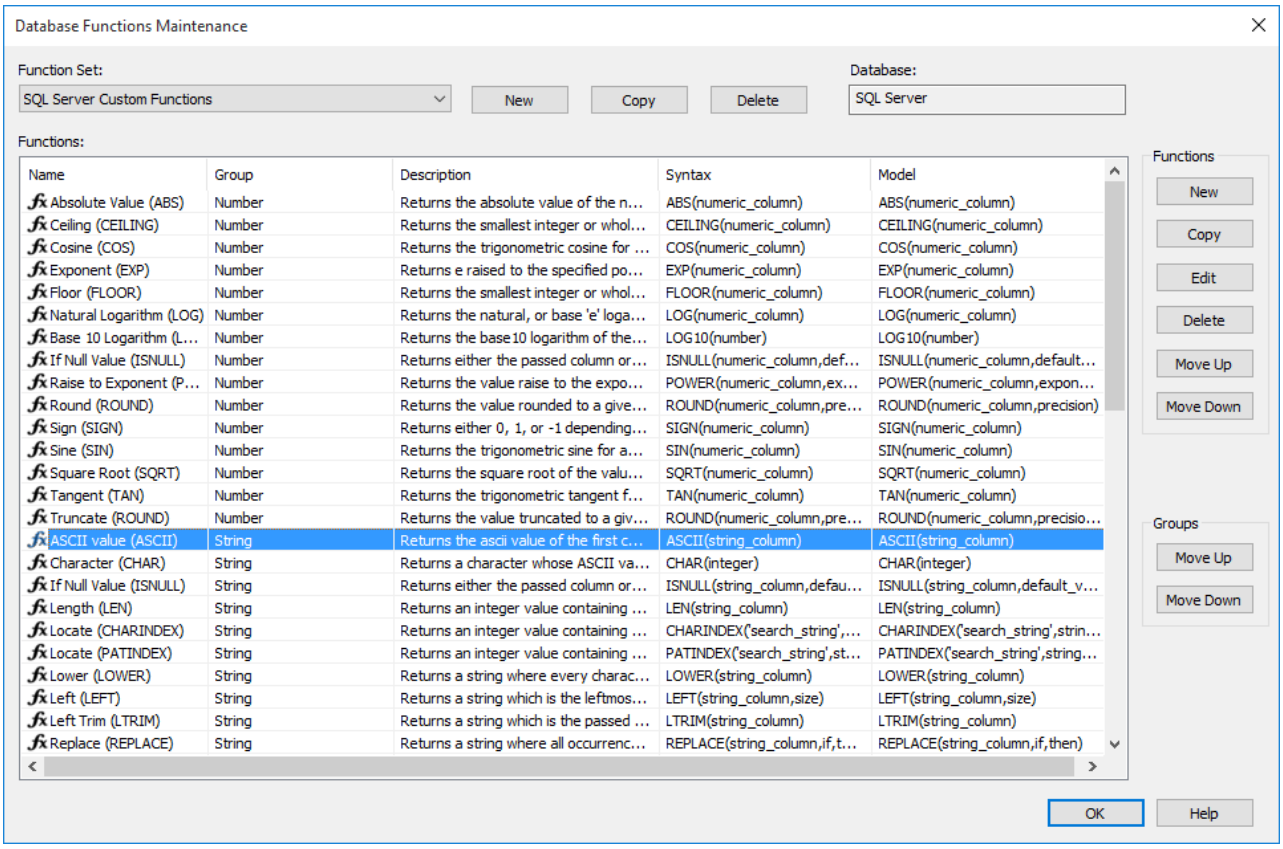
Select a function and then click the **Move Up** button on the right to move the function up in the function list, within its group; else click the **Move Down** button on the right to move the function down in the function list, within its group.



© Copyright, WhereScape Software Ltd All rights reserved.

To move a group of functions in the database function set up or down in the list:

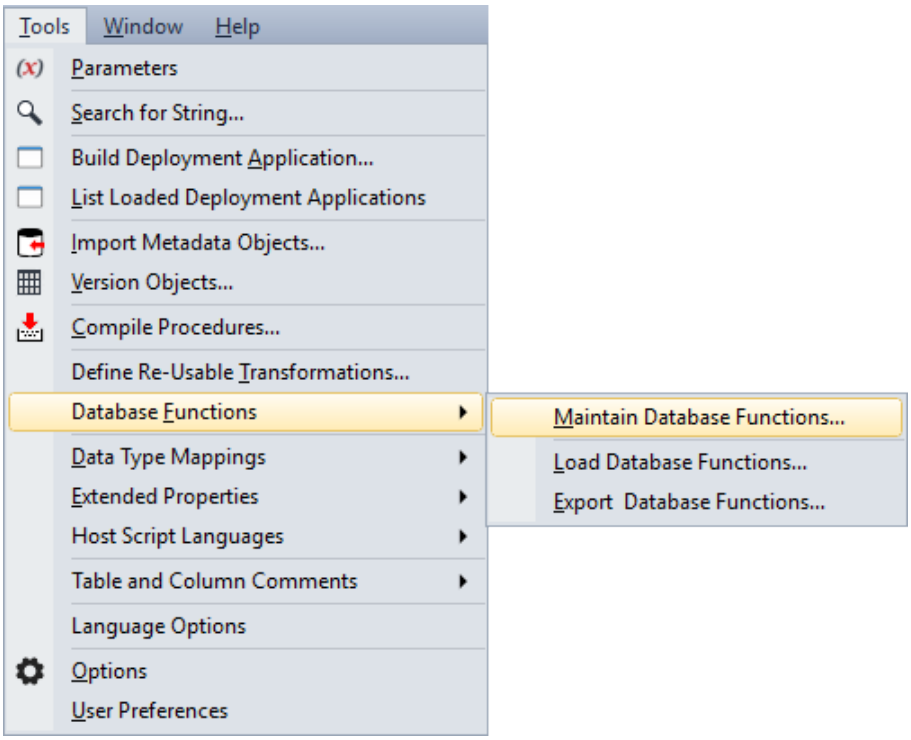
Under the **Group** column, select any function in a particular group and then click the **Move Up** button under the **Groups** heading on the right to move the function group up in the function list. Similarly, use the **Move Down** button under the Groups heading to move a function group down in the function list.



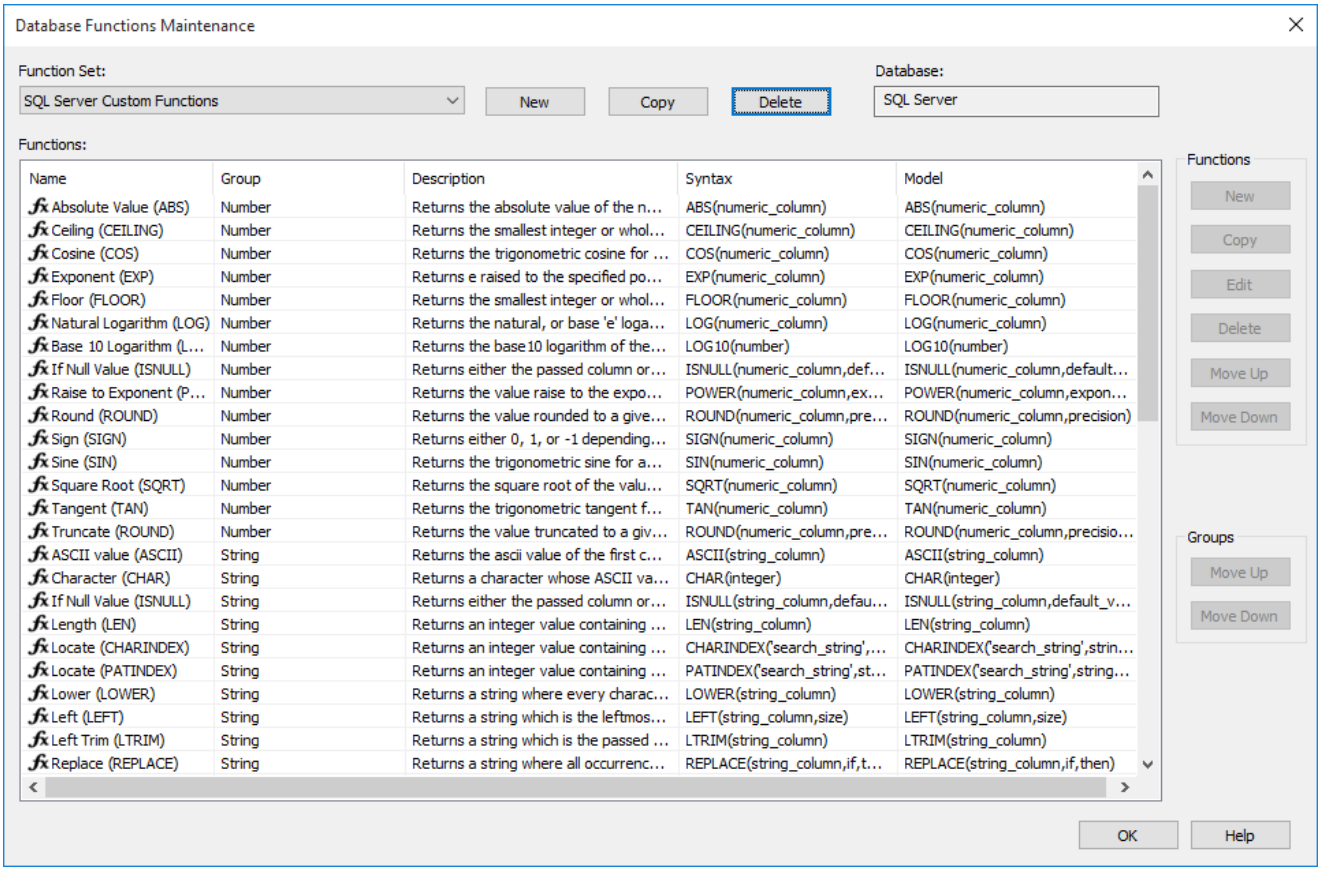
© Copyright, WhereScape Software Ltd All rights reserved.

Deleting a Database Function Set

To delete a database function set, select **Tools > Database Functions > Maintain Database Functions**.



Select a **Function Set** from the drop-down list and then click the **Delete** button.

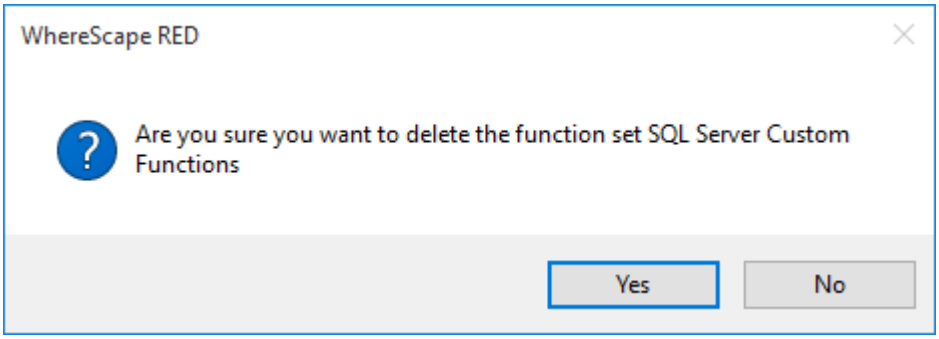


Note

The **Delete** button is not available for standard function sets.

© Copyright, WhereScape Software Ltd All rights reserved.

Click **Yes** to confirm.

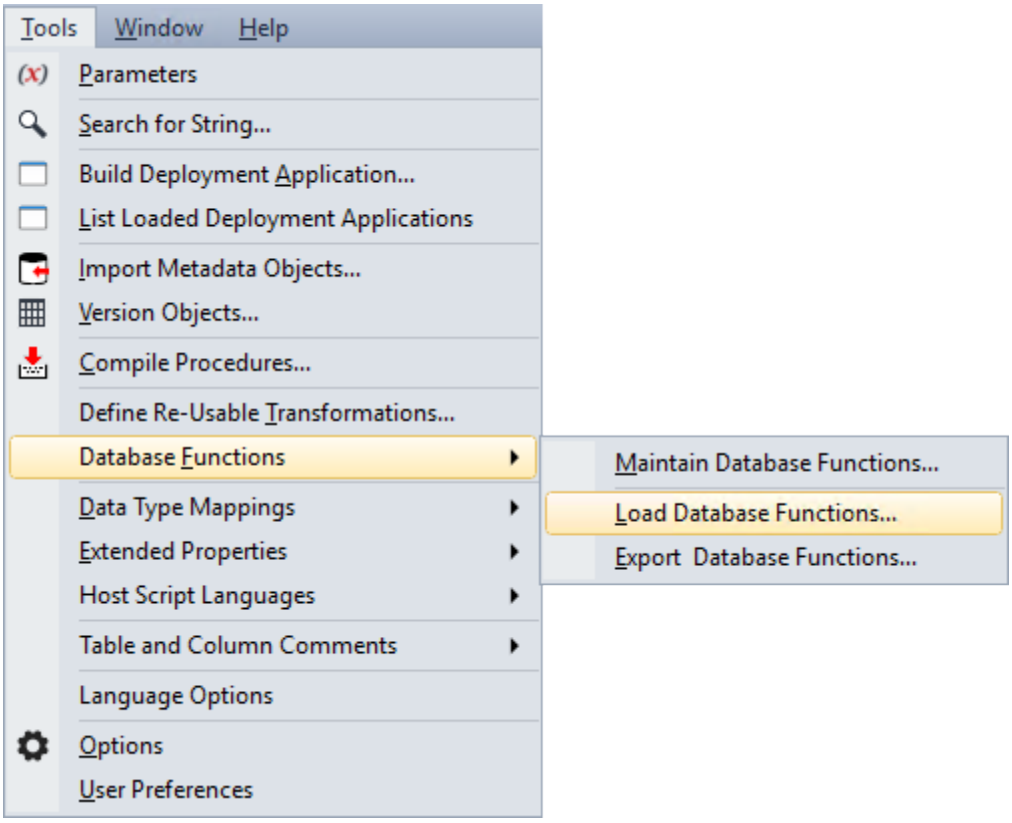


When all functions are deleted, the function set ceases to exist.

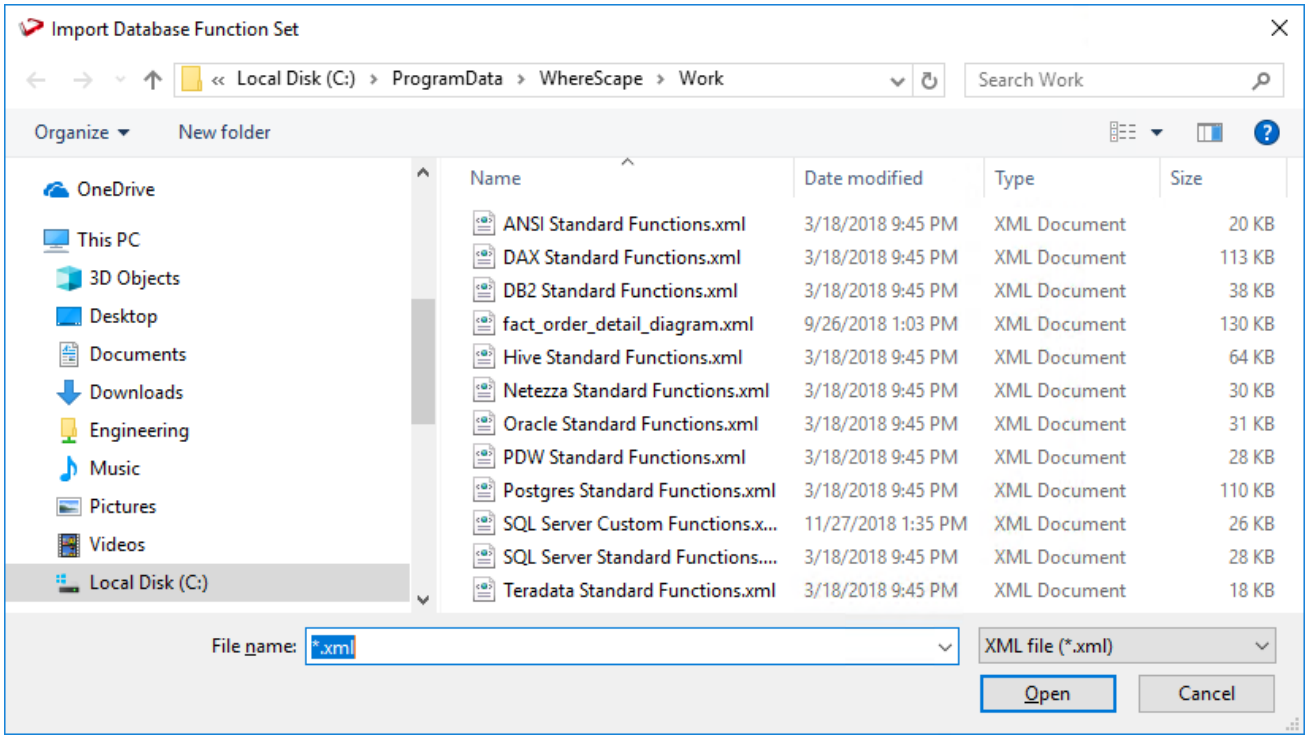
© Copyright, WhereScape Software Ltd All rights reserved.

Loading Database Function Sets

To load a database function set, select **Tools > Database Functions > Load Database Functions**.



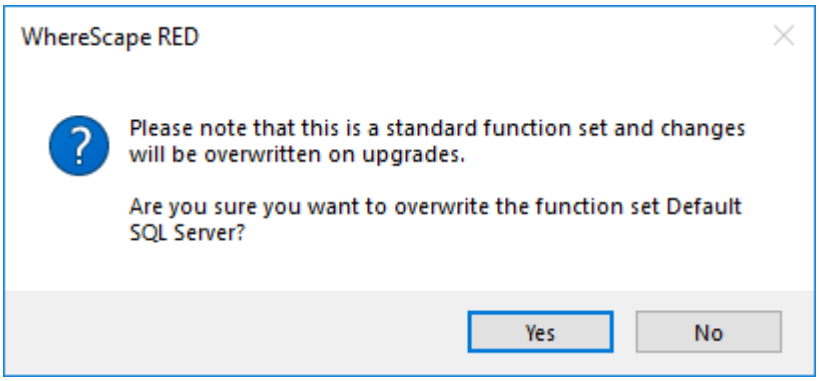
The following screen is displayed. Select the xml file to load the database functions. By default, RED expects the xml files to be in **ProgramData\WhereScape\Work** directory.



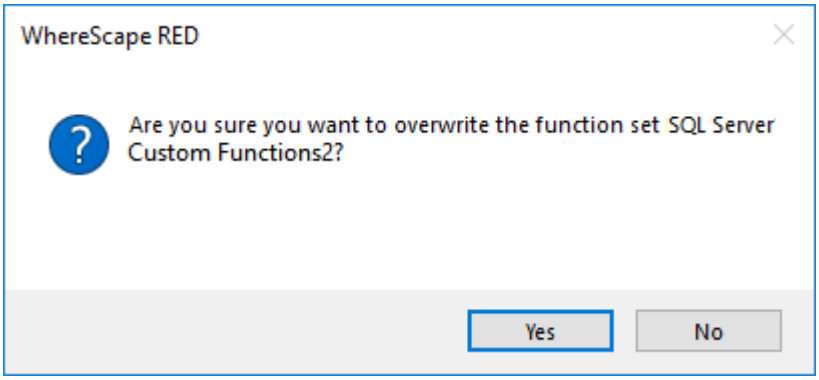
The xml file is validated using the schema definition file at <install directory>\Administrator\Function Sets\Database Function Set.xsd

© Copyright, WhereScape Software Ltd All rights reserved.

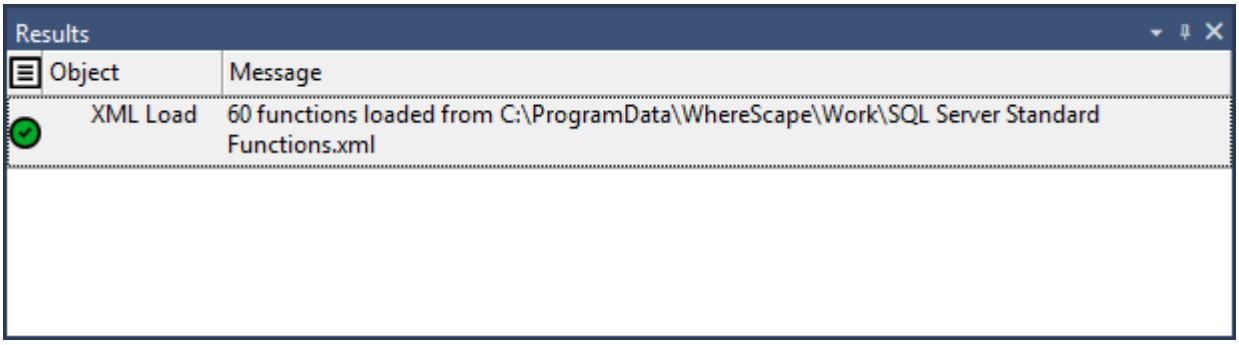
If a function set containing the phrase **Standard Function** is loaded, a warning is displayed:



If an existing function set (not a standard set) is about to be overwritten, a warning is displayed:



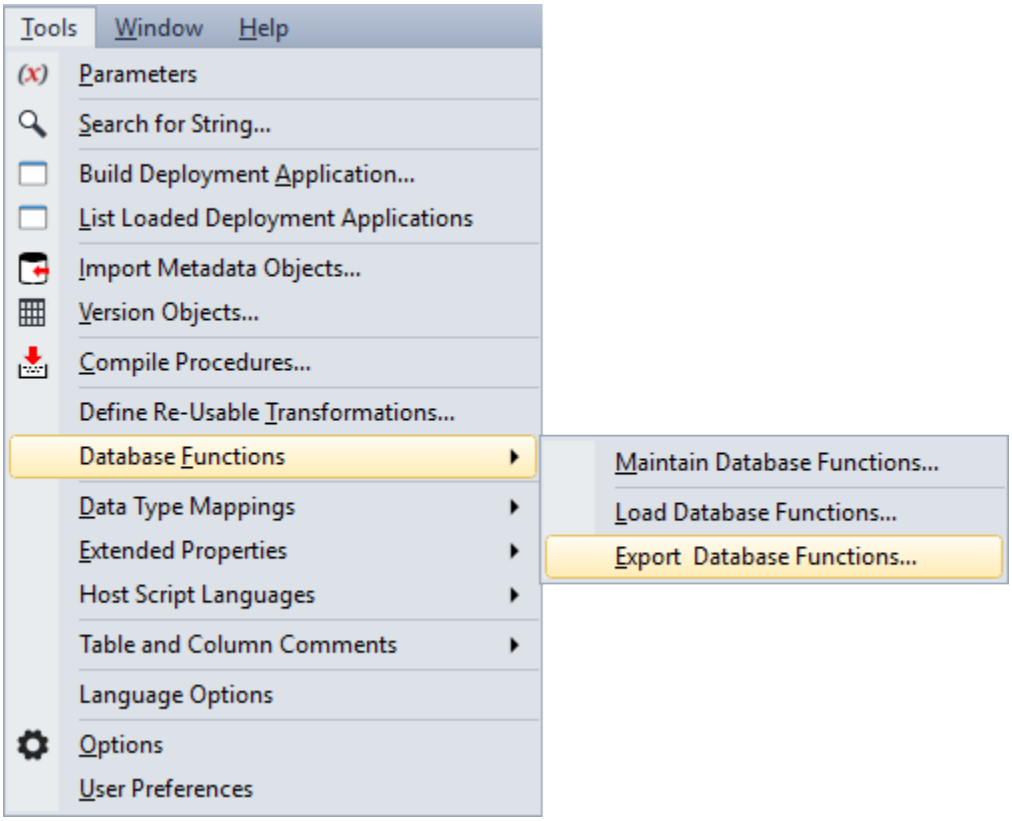
A message is displayed in the **Results** pane.



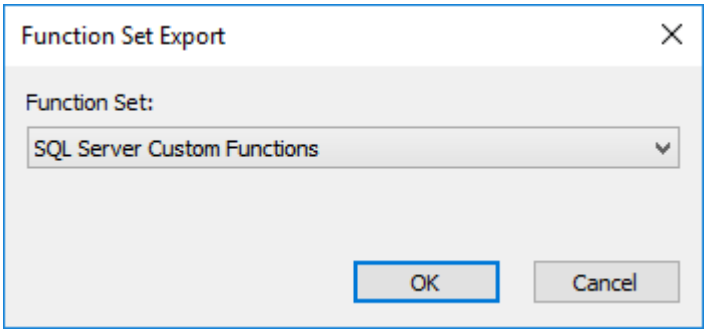
© Copyright, WhereScape Software Ltd All rights reserved.

Exporting Database Function Sets

To export Database Function Sets, select **Tools > Database Functions > Export Database Functions**.

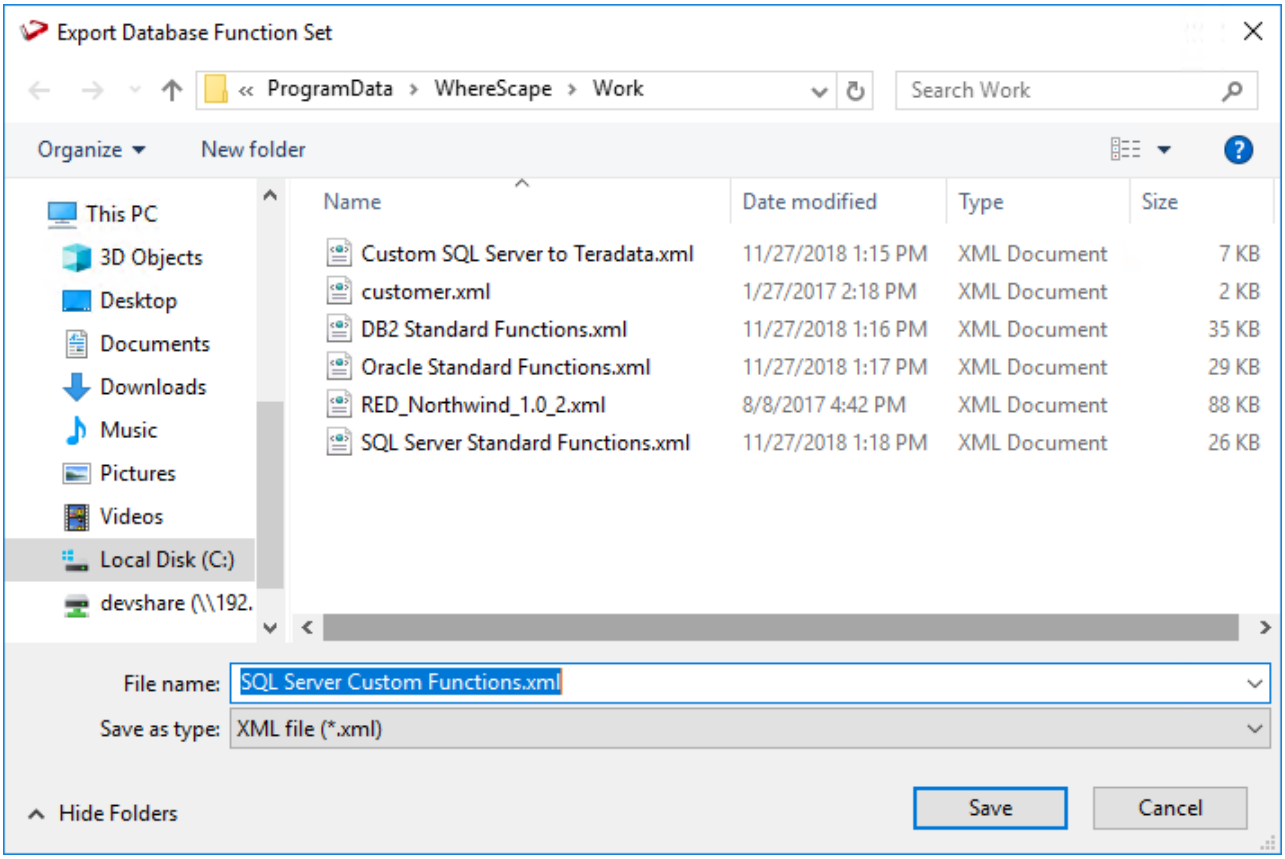


Select the **Function Set** to export from the drop-down list and then click **OK**.

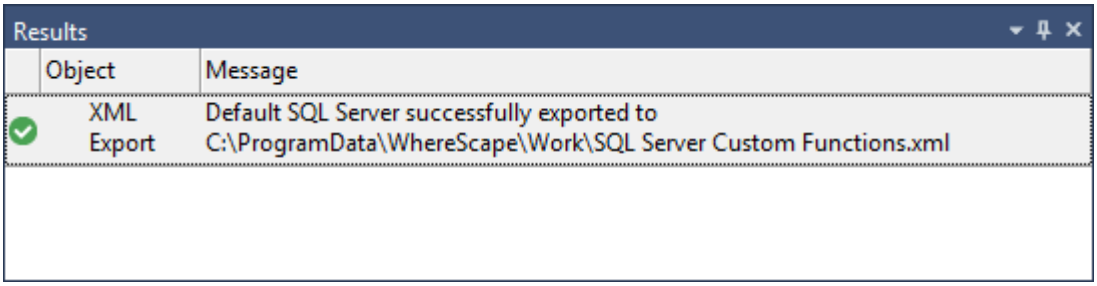


© Copyright, WhereScape Software Ltd All rights reserved.

By default, RED exports the xml file to the **ProgramData\WhereScape\Work** directory but this can be changed. Change the File name if necessary and then click **Save**.



A message is displayed in the **Results** pane.



© Copyright, WhereScape Software Ltd All rights reserved.

Extended Properties

WhereScape RED provides a facility for defining and managing extended database properties which can be used in a range of scenarios, one of the motives for their introduction was to aid interaction with the variety of new database technologies that RED customers want to exploit for their Business Intelligence infrastructure

To support these emerging database technologies, WhereScape provides enablement packs for a variety of target “custom” database types, which includes Snowflake, Amazon Redshift and Microsoft Azure SQL data warehouse.

One component of these packs are a set of extended properties to facilitate the connection to a specific database technology.

This feature also supports importing a set of extended property definitions into RED from an external source, and exporting extended properties defined in RED to a file. This enables you to migrate and reuse extended property definitions between different repositories.

Using the extended properties feature involves defining the extended properties to specify the variables and their scope; and then assigning values to the defined extended properties.

Extended Property Definitions

Extended property definitions primarily consist of a unique name and a scope (database type and object type), which specifies where the extended property will be available.

The extended property definitions are created and maintained in the **Extended Properties Maintenance** screen.

Extended Property Value Assignment

Once an extended property has been defined for a connection or table object, users can then assign values to each of the variable attributes available in the extended property.

The extended property values are specified and maintained in the **Extended Properties** settings in the Connection or Table object's **Properties** screen.

Extended Property Lookup

The extended property variables and their assigned values are referenced in PowerShell scripts. PowerShell script based processing is available for all objects in a custom database target. However, in a non-custom database target; PowerShell scripts are only available to process Load and Export objects.

WhereScape provides a specialized enablement pack for custom database customers for their database platform, which includes a utility template with extended property lookup functions. For more information, please contact your WhereScape representative.

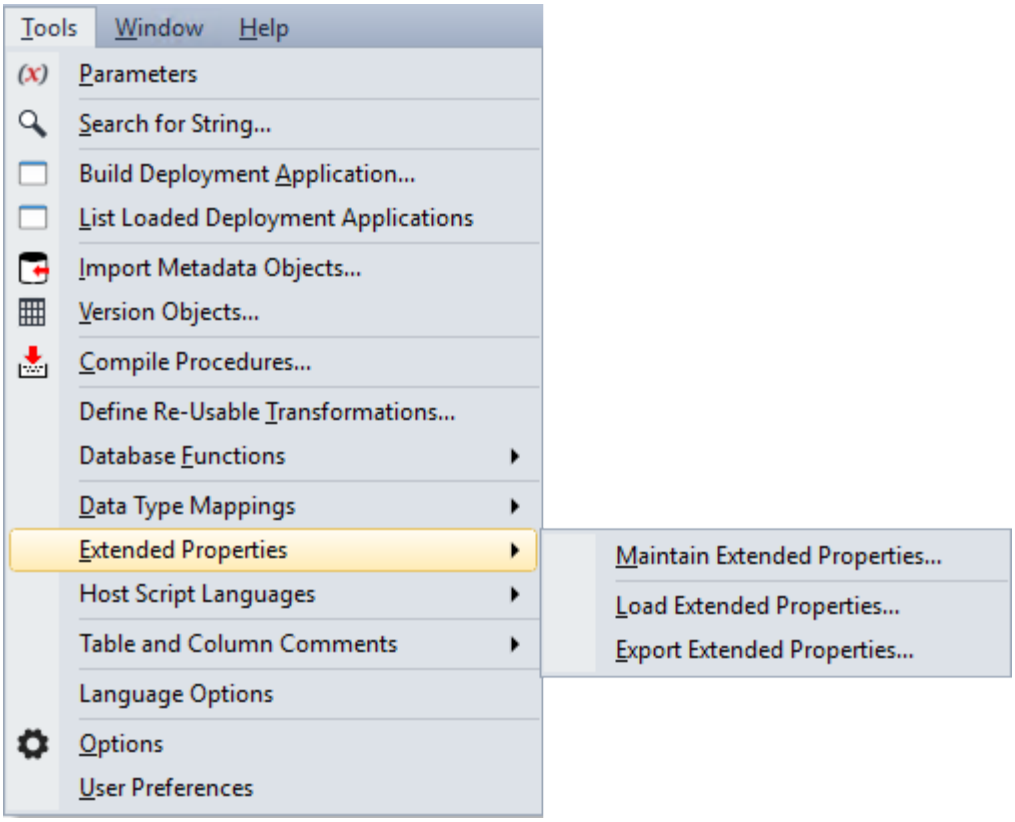
© Copyright, WhereScape Software Ltd All rights reserved.

Defining Extended Properties

An extended property definition mainly consists of the following attributes:

- **Variable** - the name of the extended property in the format it is accessed from RED and included into scripts.
- **Scope** - the scope is a sub-set definition of an extended property variable which specifies the subset of database types and objects that will have access to the variable.

The user defined extended properties can be created and managed, using the **Extended Properties** facility which is launched from the **Tools** menu.



The following options are available:

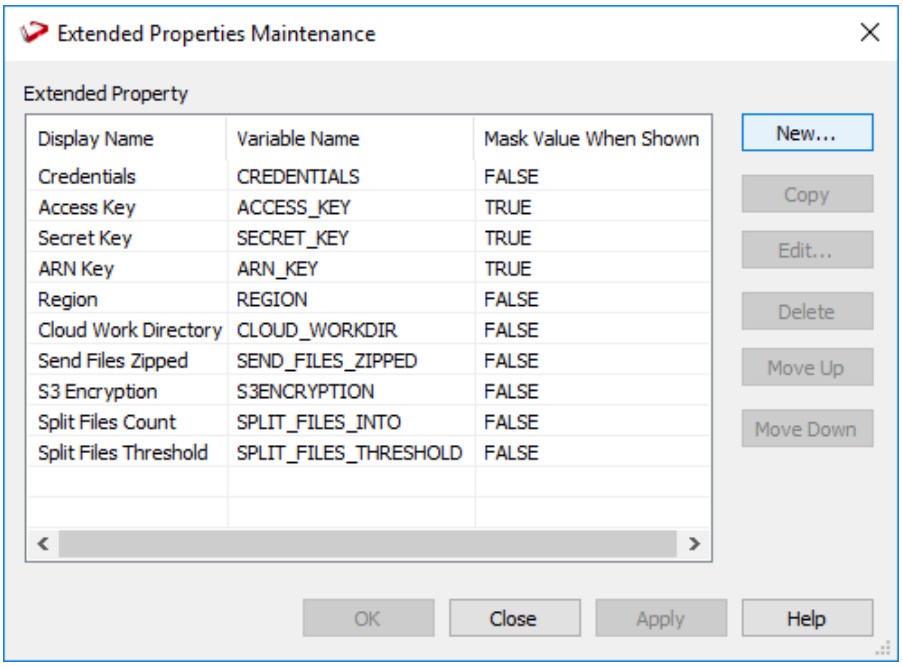
- **Maintain Extended Properties** is used to add new or edit existing extended property definition settings.
- **Load Extended Properties** is used to load a set of extended property definitions into RED from an external source. This option is typically used with a WhereScape provided enablement pack.
- **Export Extended Properties** is used to export the available extended properties to an external properties file (.extprop).

© Copyright, WhereScape Software Ltd All rights reserved.

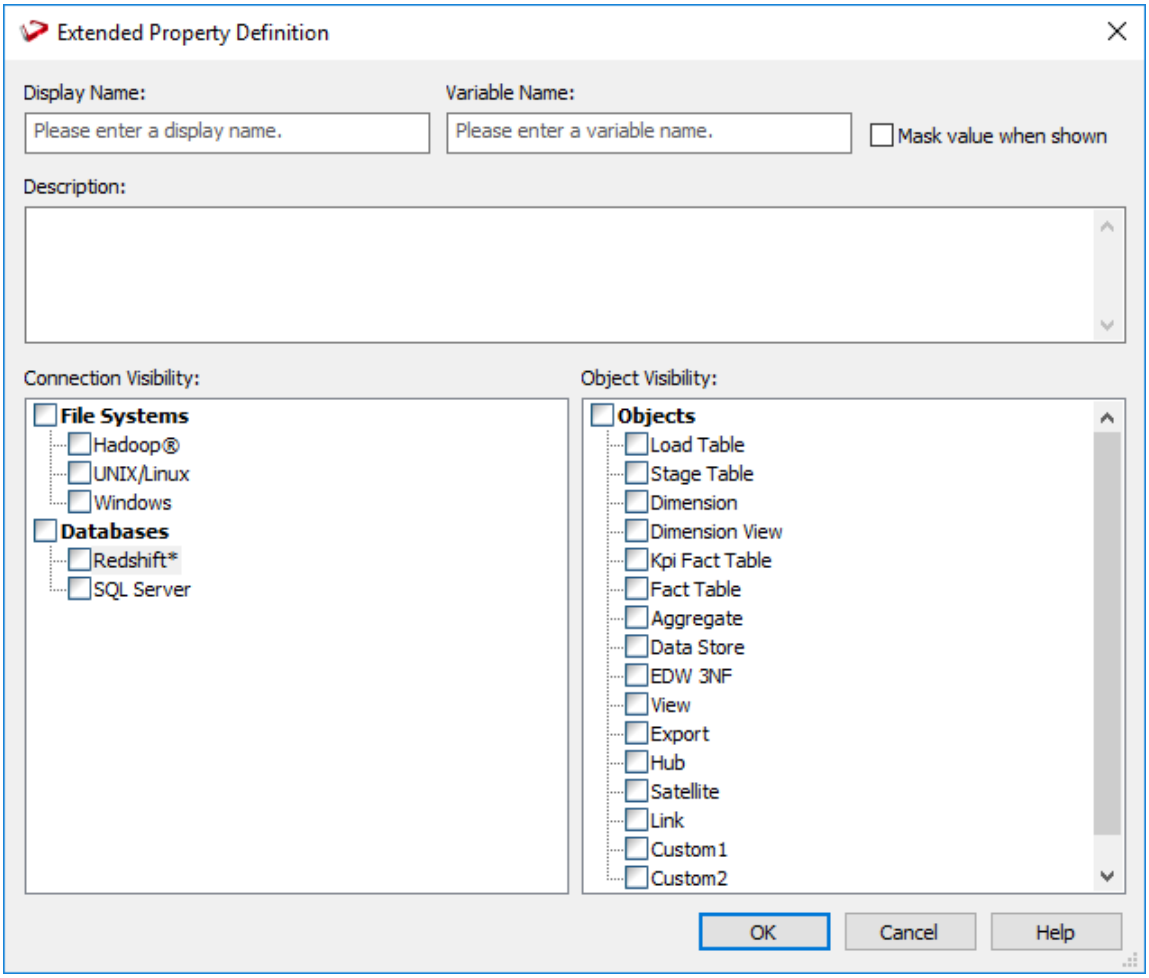
Creating an Extended Property Definition

To create a new extended property definition, select **Tools > Extended Properties > Maintain Extended Properties**.

On the **Extended Properties Maintenance** screen, click **New**.



On the **Extended Property Definition** screen, enter the display and variable names and then configure the settings as described below.



© Copyright, WhereScape Software Ltd All rights reserved.

Variable Settings

Menu Options	Description
Display Name	Enter a display name for the variable. This is the extended property name displayed in the Connection or Table properties.
Variable Name	Enter a unique variable name in the format it is accessed from the RED metadata service and included into scripts.
Mask Value when shown	Select this check box if you want to mask the value of the extended property, when displayed in the Connection or Table properties. The extended property value is also masked in the RED generated technical documentation. **Note** During an application creation process (via the **Tools > Build Deployment Application** menu), any extended property variables/assignments applied to a connection or table object are included in the generated application files. However, a connection or table object extended property with a masked value will not be included in the generated application files.
Description	Enter a description for the variable and its purpose/use

Scope Settings

Menu Options	Description
Connection Visibility	Enables you to control the scope of the extended property definition, by selecting a Connection **File System** and/or a **Database** type that can access the extended property variable and their assigned value.For example, if only the SQL Server check box is selected under the **Databases** option, then the variable will only appear in the Extended Properties screen of connections with an SQL Server database type and not in the Extended Properties screen of any other connections that is not an SQL Server database.
Object Visibility	Enables you to control the scope of the extended property definition, by selecting an **Object** type that can access the extended property variable and their assigned value. For example, if only the **Load Table** check box is selected under the **Objects** option, then the variable will only appear in the Extended Properties screen of Load tables and not in the Extended Properties screen of any other table objects.

Notes:

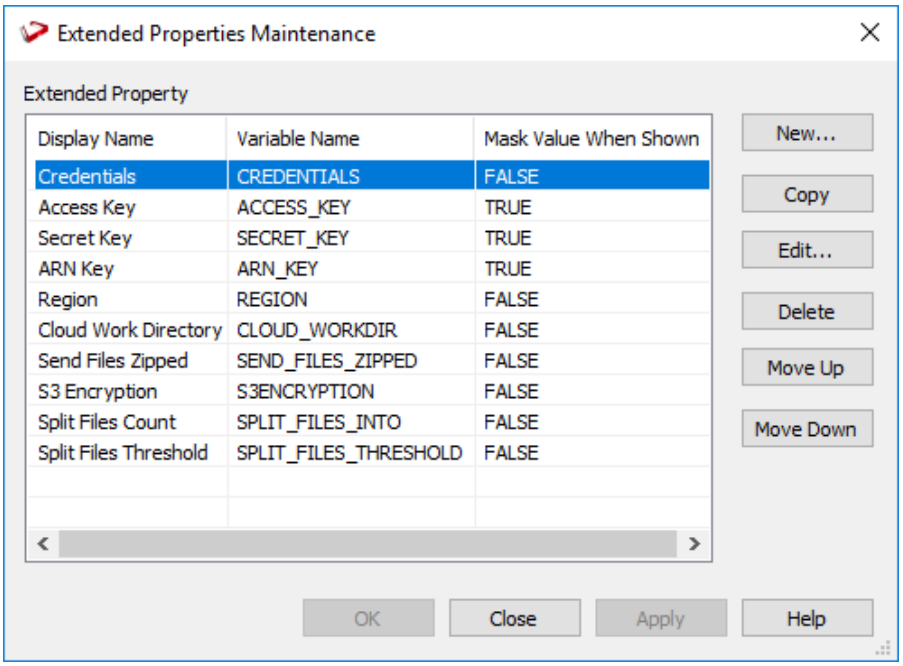
- Extended property variables can be set for index objects. An extended property's scope is still defined by object type but the value for the variable can be set for the table and each index independently.
- Automatically created indexes are recreated each time the table's procedure is rebuilt. Therefore, the values set for extended properties on automatically created indexes are lost when they are recreated. Manually created indexes and their extended properties are persisted.
- As with extended properties defined for tables and connections, extended properties defined on indexes can be accessed from templates, including create index DDL, drop index DDL and update procedure/script.

© Copyright, WhereScape Software Ltd All rights reserved.

Maintaining Extended Property Definitions

The **Extended Properties Maintenance** screen displays a list of existing extended property definitions and enables you to add a **New** extended property definition. Clicking an extended property definition from the list enables you to **Copy**, **Edit** or **Delete** the selected extended property.

In addition, you can change the position of a selected extended property definition by moving it **Up** or **Down** within the list to specify a preferred order or grouping arrangement, when the extended property definitions are displayed in the Connection or Table properties screens. The display order has no impact upon the operation.



The following describe the elements of the **Extended Properties Maintenance** screen.

Extended Property

This pane lists the extended property definitions that have been setup in RED. It details the display and variable name and indicates if the value of the extended property is masked, when displayed in the Connection or Table properties.

Function Buttons

These buttons enable you to create and manage the extended property definitions.

Button	Description
New	Launches the **Extended Property Definition** screen which enables you to create a new variable and specify the subset of database types and objects that will have access to the variable. Refer to **Creating an Extended Property Definition** for details.
Copy	Duplicates the selected extended property and adds a **_Copy1** suffix to the **Display Name** and **Property Name**. All fields are populated with the values of the original extended property. These can be edited as required.
Edit	Launches the **Extended Property Definition** screen which enables you to edit the selected extended property.
Delete	Deletes the selected extended property, if it is not currently in use. The user is warned if it is in use and asked to confirm or cancel the delete action.
Move Up/Move Down	Moves the selected extended property Up or Down the list. The display order of the extended property has no impact on its use.

Notes

- If an extended property is deleted, both the definition and all corresponding value assignments are

© Copyright, WhereScape Software Ltd All rights reserved.

Notes

deleted from the meta data repository. It is removed from the **Extended Properties** settings of the Connection or Table object Properties screen.

- If an extended property definition variable name is changed, the change is applied in the **Extended Properties** settings of the Connection or Table object Properties screen and its corresponding value assignment is retained.
- If an extended property definition scope is changed, its variable and corresponding value assignments are not deleted from the meta data repository. It is hidden from the **Extended Properties** settings of the Connection or Table object Properties screen. For example, an extended property variable has the Load object type set as in-scope and a corresponding value assignment has also been set in the Extended Properties setting in the Table Properties screen.
  If the extended property definition is edited to remove the Load object type from its scope, then the value assignment is not removed but becomes inactive and RED displays a warning that the scope change impacts a value assignment.

Extended Properties Value Assignment

After defining the extended property, you can assign values to each of the variable available in the **Extended Properties** tab of in-scope objects.

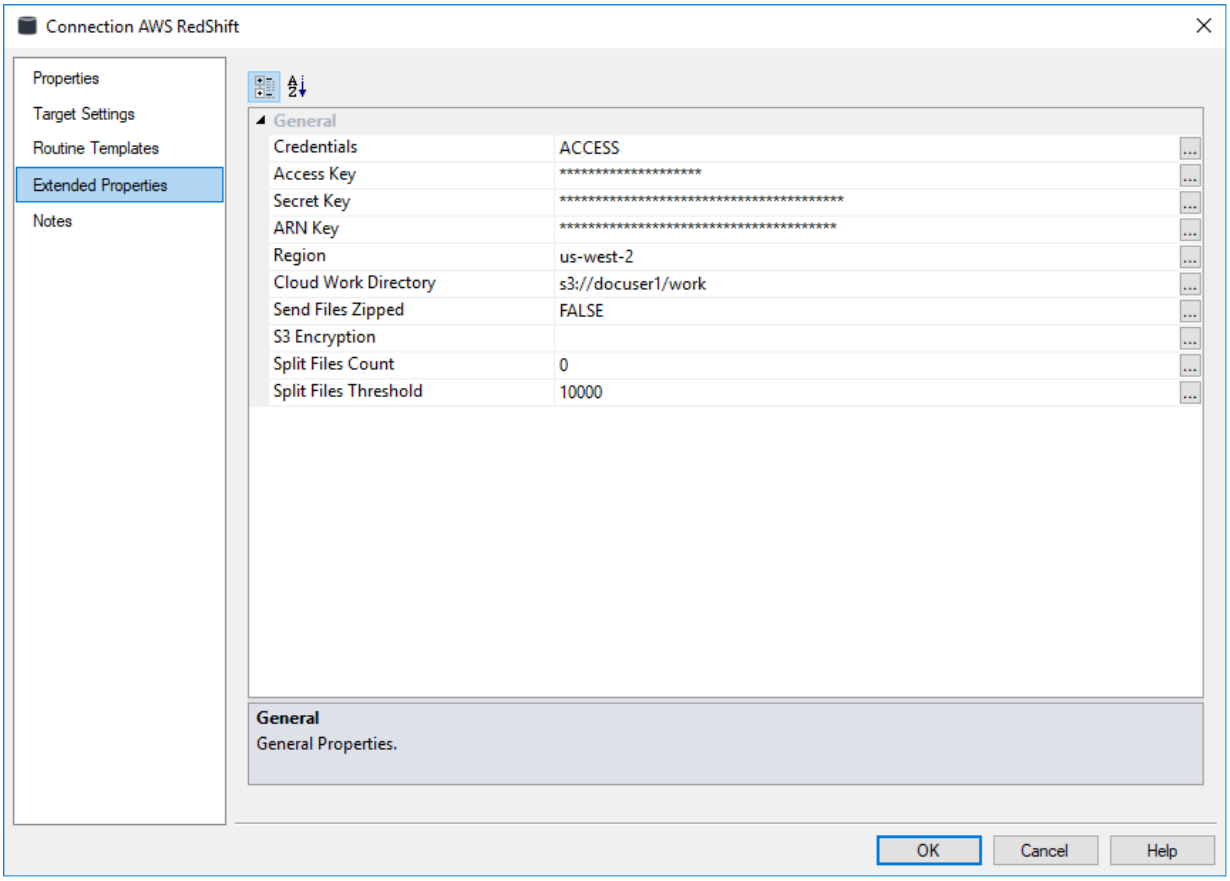
One extended property can have several connections and objects in its scope and therefore can have several values and assignments—one per connection and object.

An extended property assignment establishes the relationship between a variable and its value, which is specific for a particular connection or object.

Setting Up Extended Property Values for a Connection

The extended property values for a connection are assigned and maintained in the **Extended Properties** tab of the connection **Properties** screen.

Connection Extended Properties

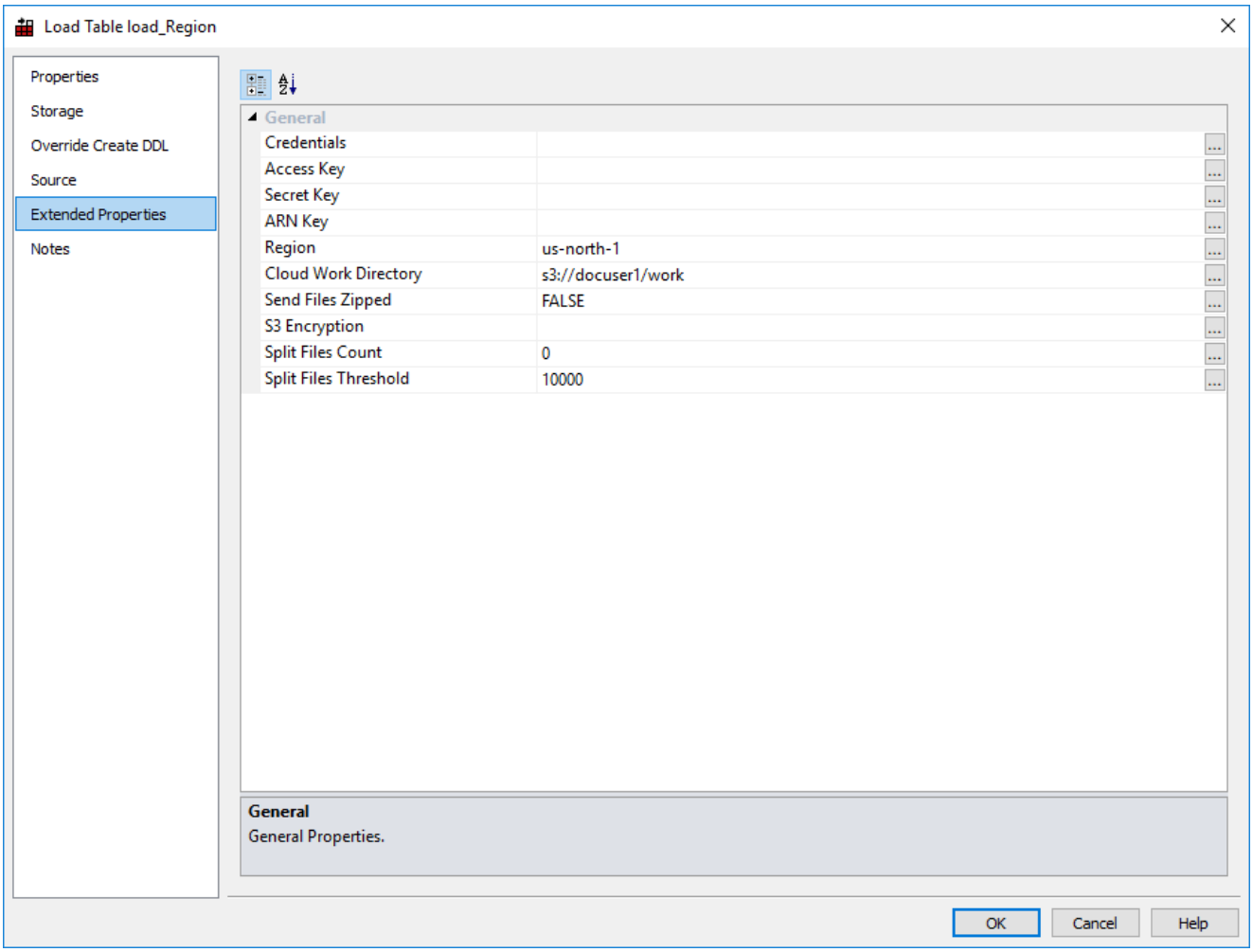


© Copyright, WhereScape Software Ltd All rights reserved.

Setting Up Extended Property Values for an Object

The extended property values for an object are assigned and maintained in the **Extended Properties** tab of the table **Properties** screen.

WhereScape RED Object Extended Properties



Extended Properties Lookup

The extended property variables and their assigned values are referenced in PowerShell scripts. PowerShell script based processing is available for all objects in a custom database target. However, in a non-custom database target; PowerShell scripts are only available to process Load and Export objects.

WhereScape provides a specialized enablement pack for custom database customers for their database platform, which includes a utility template with extended property lookup functions. For more information, please contact your WhereScape representative.

© Copyright, WhereScape Software Ltd All rights reserved.

Extended Properties Data Migration Between Repositories

Once extended properties are defined and set, they can be propagated to the other RED repositories as follows:

The definitions are exported from the source repository and imported to the target repository via the **Tools > Extended Properties** menu.

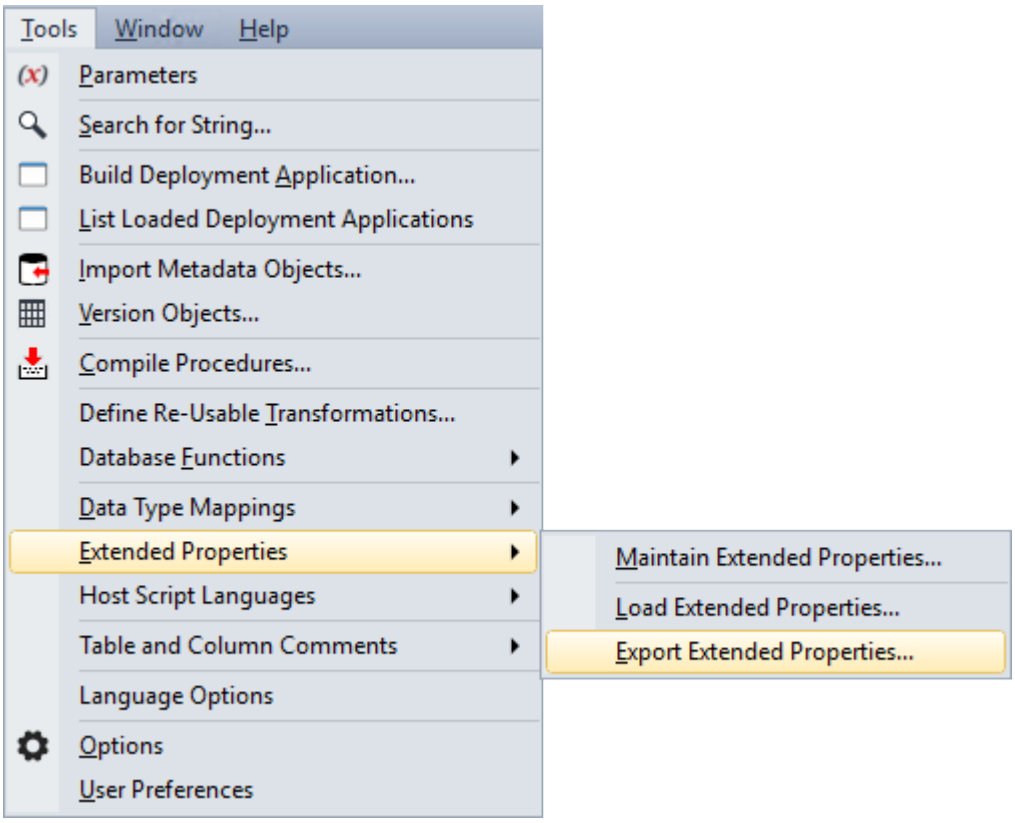
The values assigned to extended properties are included in a RED application that includes the parent connection or table object and are imported to the target repository during the application load process.

Notes:

1. Any new or changed extended property definition must first be exported and then imported into the downstream repository, prior to loading an application that references them. Failing to load the definition first will result in the value assignment not being loaded for the object.
2. Extended properties are designed to be accessed from Windows PowerShell scripts, via the WhereScape RED metadata service. To aid this access, WhereScape provides a metadata service client library that customers are suggested to use, to access the metadata service.

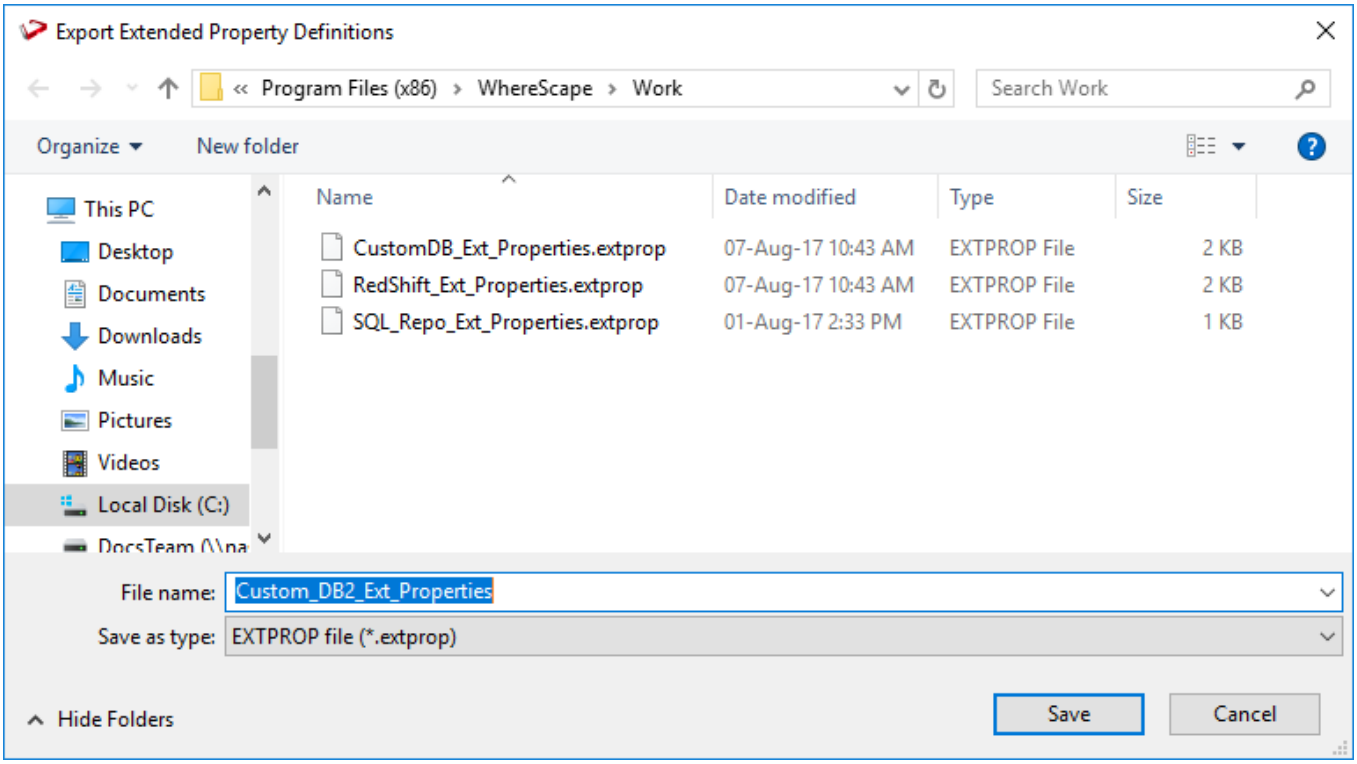
Exporting Extended Properties

To export extended properties from RED, select **Tools > Extended Properties > Export Extended Properties**.

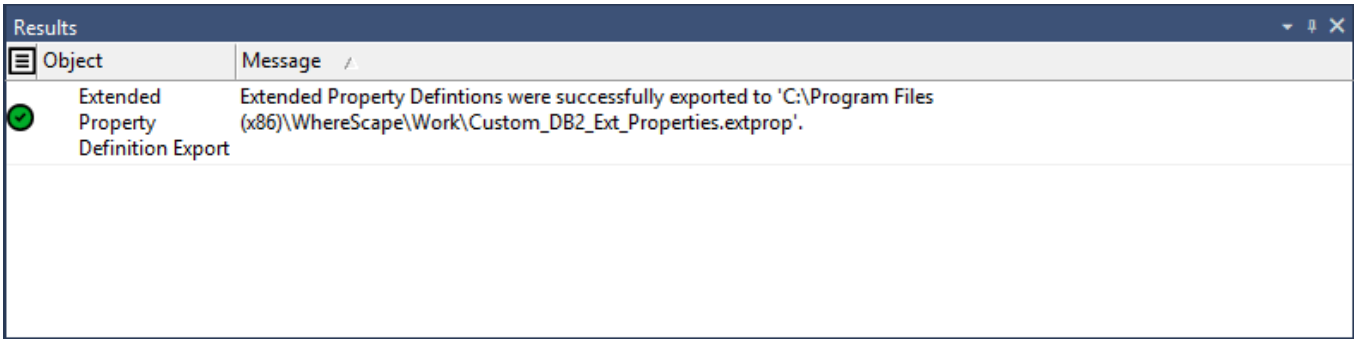


© Copyright, WhereScape Software Ltd All rights reserved.

The following window is displayed. By default, RED exports the extended property (EXTPROP) file to **Program Files\WhereScape\Work** directory, but this can be changed. Type in the **File name** and then click **Save**.



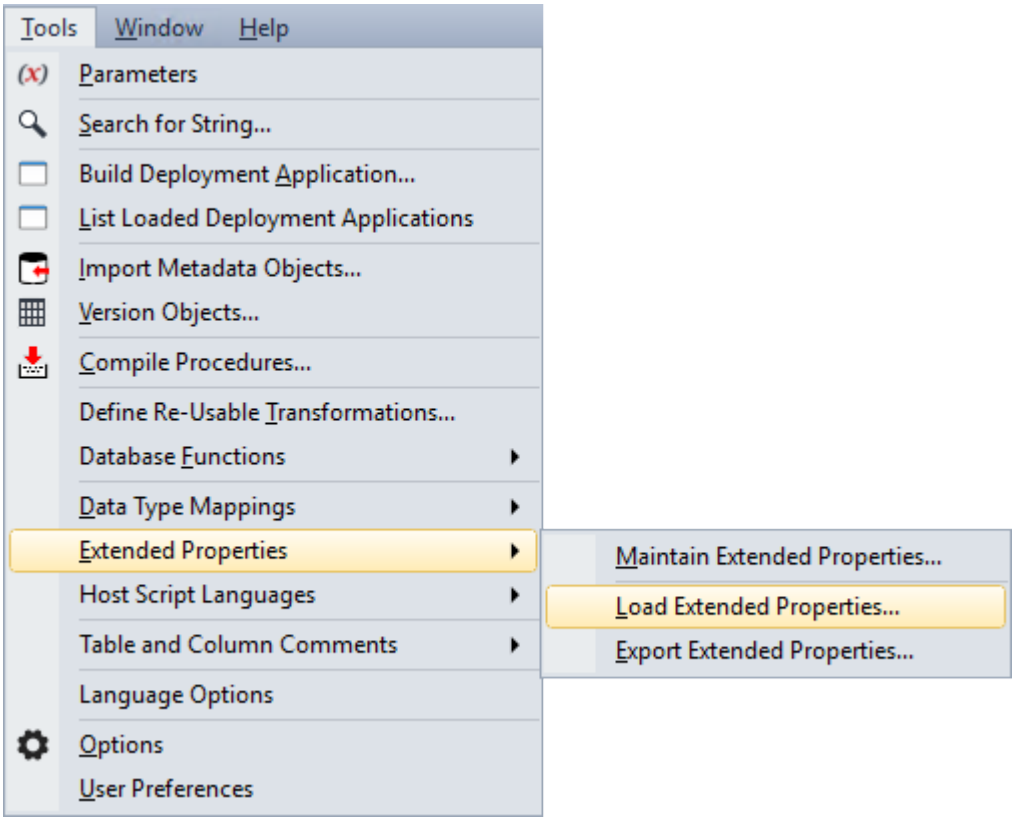
A confirmation message is displayed in the **Results** pane.



© Copyright, WhereScape Software Ltd All rights reserved.

Loading Extended Properties

To load an extended property file, select **Tools > Extended Properties > Load Extended Properties**.



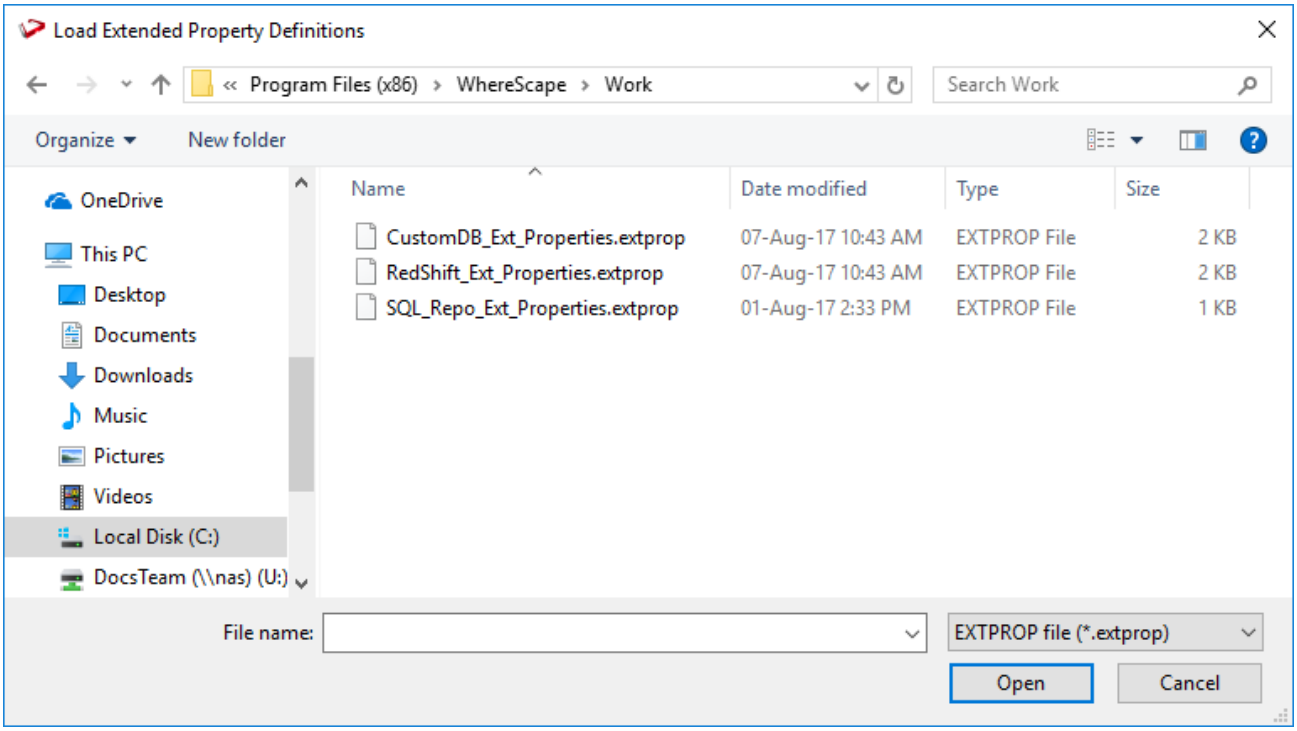
Note

The values assigned to extended properties are included in a RED application that includes the parent Connection or table Object and are imported to the target repository, during the application load process.

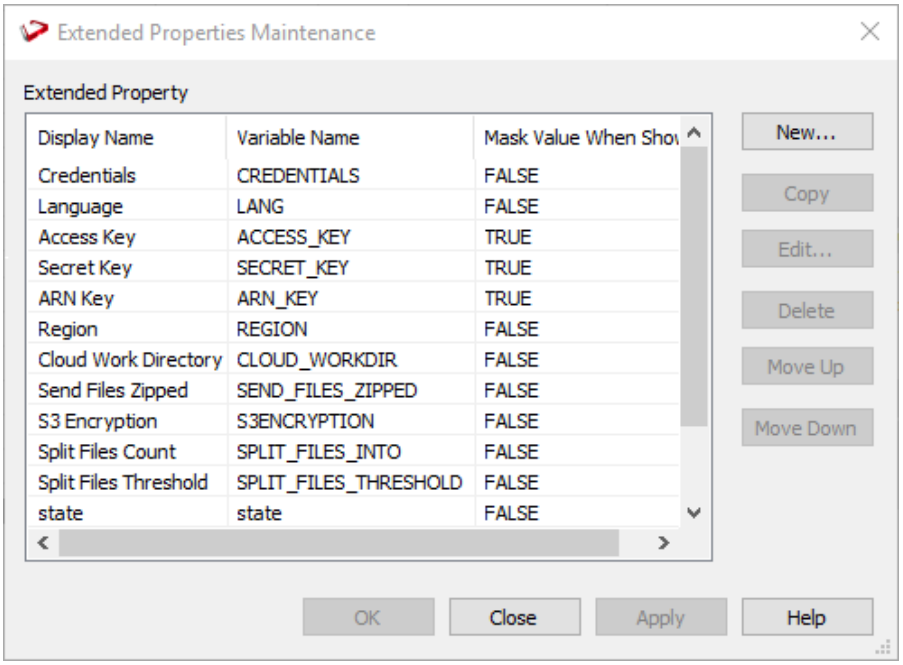
An application including an object with an extended property value set, is dependent on the extended property definition being present in the target repository.

© Copyright, WhereScape Software Ltd All rights reserved.

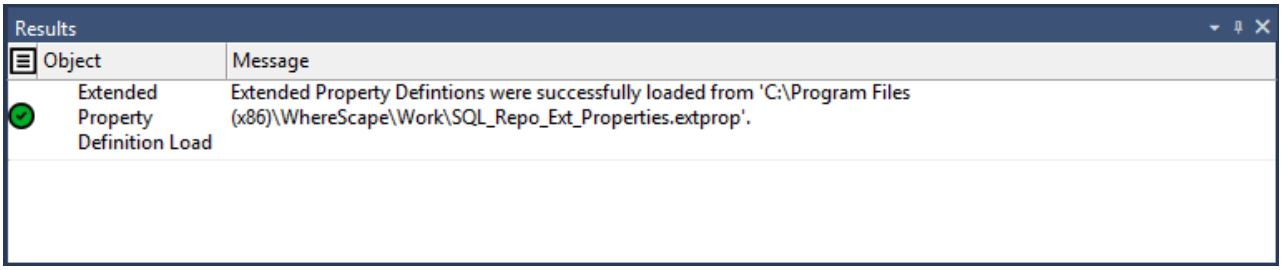
The following window is displayed. Select the extended property (EXTPROP) file to load. By default, RED expects the extprop files to be in **Program Files\WhereScape\Work** directory, but this can be changed.



The selected extprop file is loaded and the attributes are listed in the **Extended Properties Maintenance** window.



A confirmation message is displayed in the **Results** pane.



© Copyright, WhereScape Software Ltd All rights reserved.

Multi Source Processing

WhereScape RED enables you to process data from more than one source query into a single consolidated table. The multi-source processing capability is designed to provide users with flexibility in the definition of component source statements and the ability to independently process multi source tables that are associated with a target table.

Multi Source Functions and Features

The following describes the WhereScape RED functions and features that are used to support multi source processing of data warehouse objects, created in RED.

Note

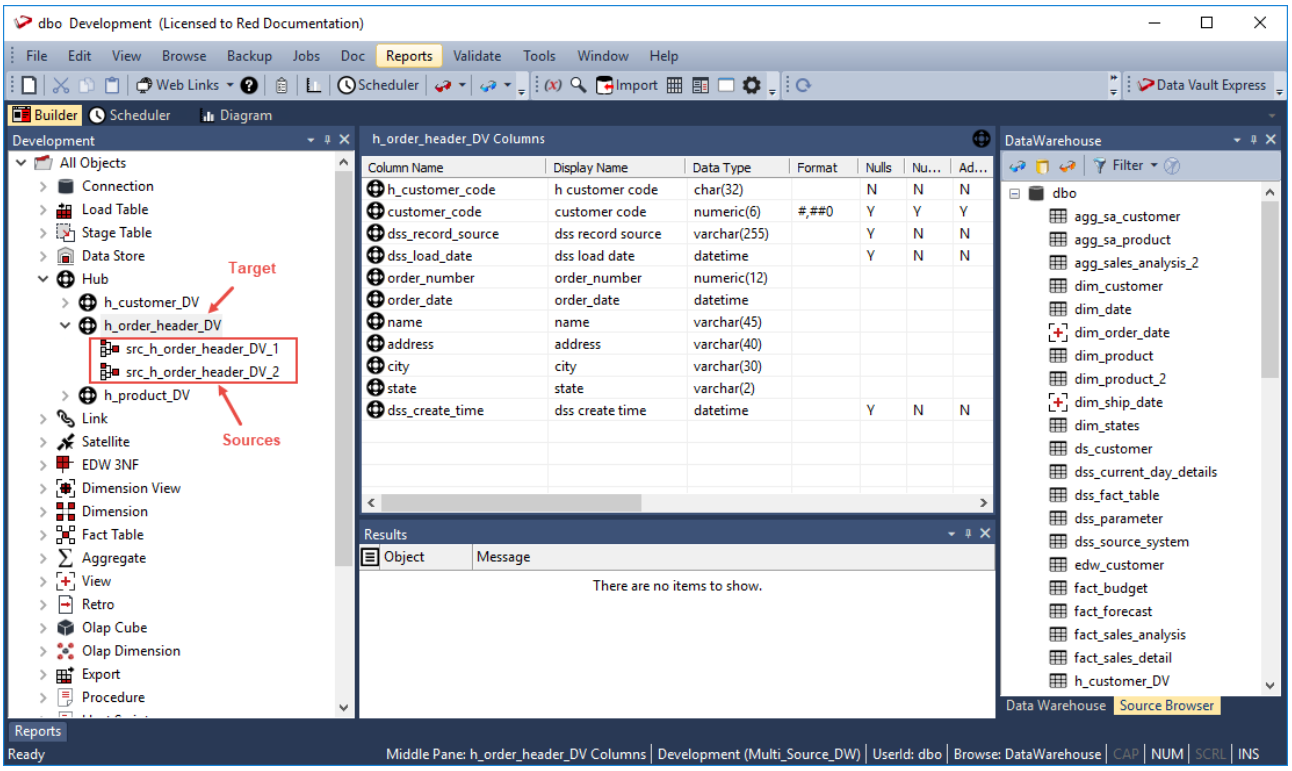
The source mapping feature can be used with all RED table objects, except for Tabular objects, partitioned tables and Load tables.

Source Mapping Object

Source Mapping objects are child objects that are used in WhereScape RED to map columns from one or more source tables to an existing target table in RED.

A Source Mapping object is built from the Data Warehouse connection and can be created from one or more source tables.

Refer to Adding Source Mapping to Objects for details.



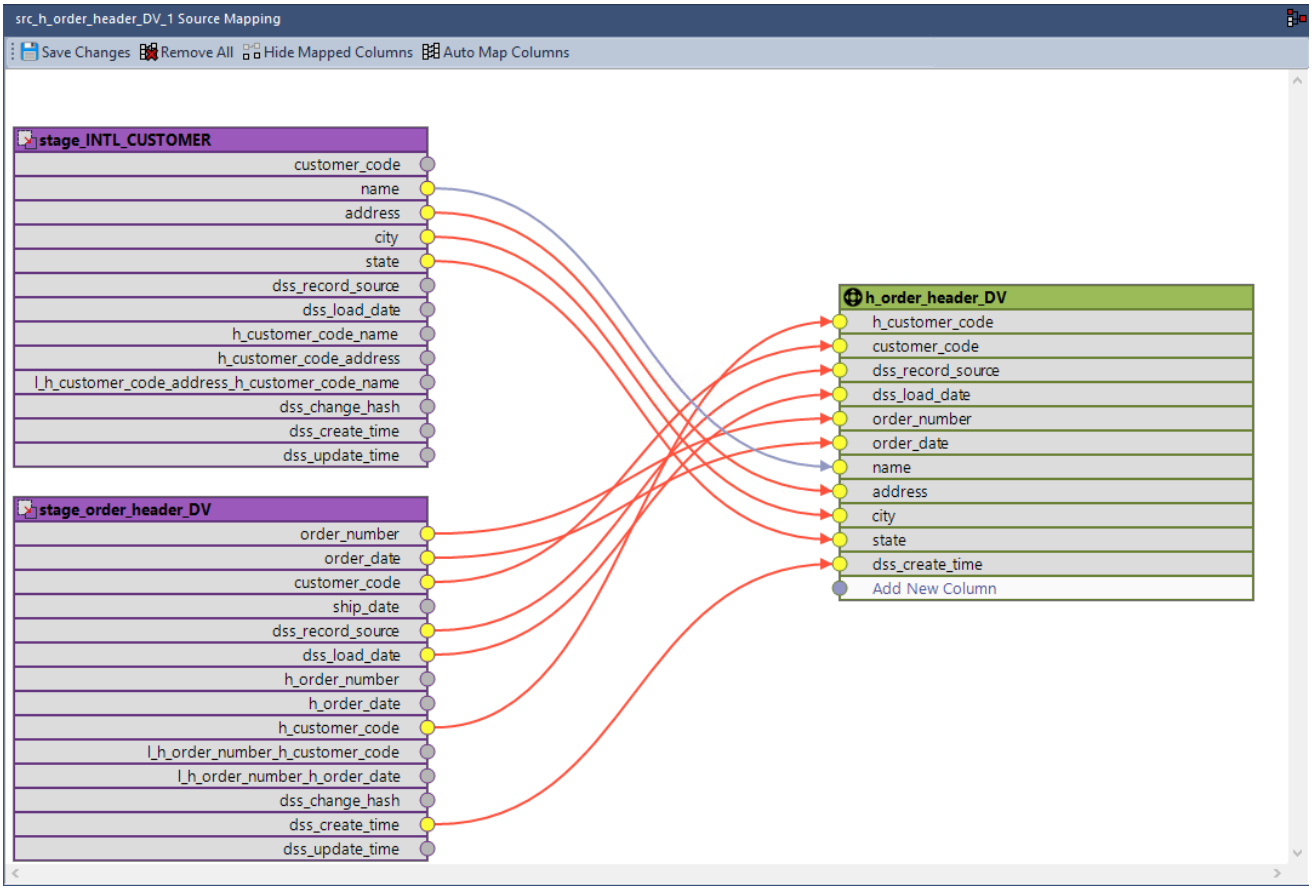
© Copyright, WhereScape Software Ltd All rights reserved.

Source Mapping Tool

A Source Mapping tool is available in WhereScape RED which enables you to graphically map columns from one or more source tables to an existing target table in RED.

This tool provides a graphical representation of the mapped columns between the source tables and the target table. The columns from the source table that exists in the target table are automatically mapped. You can edit the mappings by clicking a source column connection point and dragging the line that appears to the target column's connection point.

Refer to the section **Maintaining Source Column Mappings** for details.

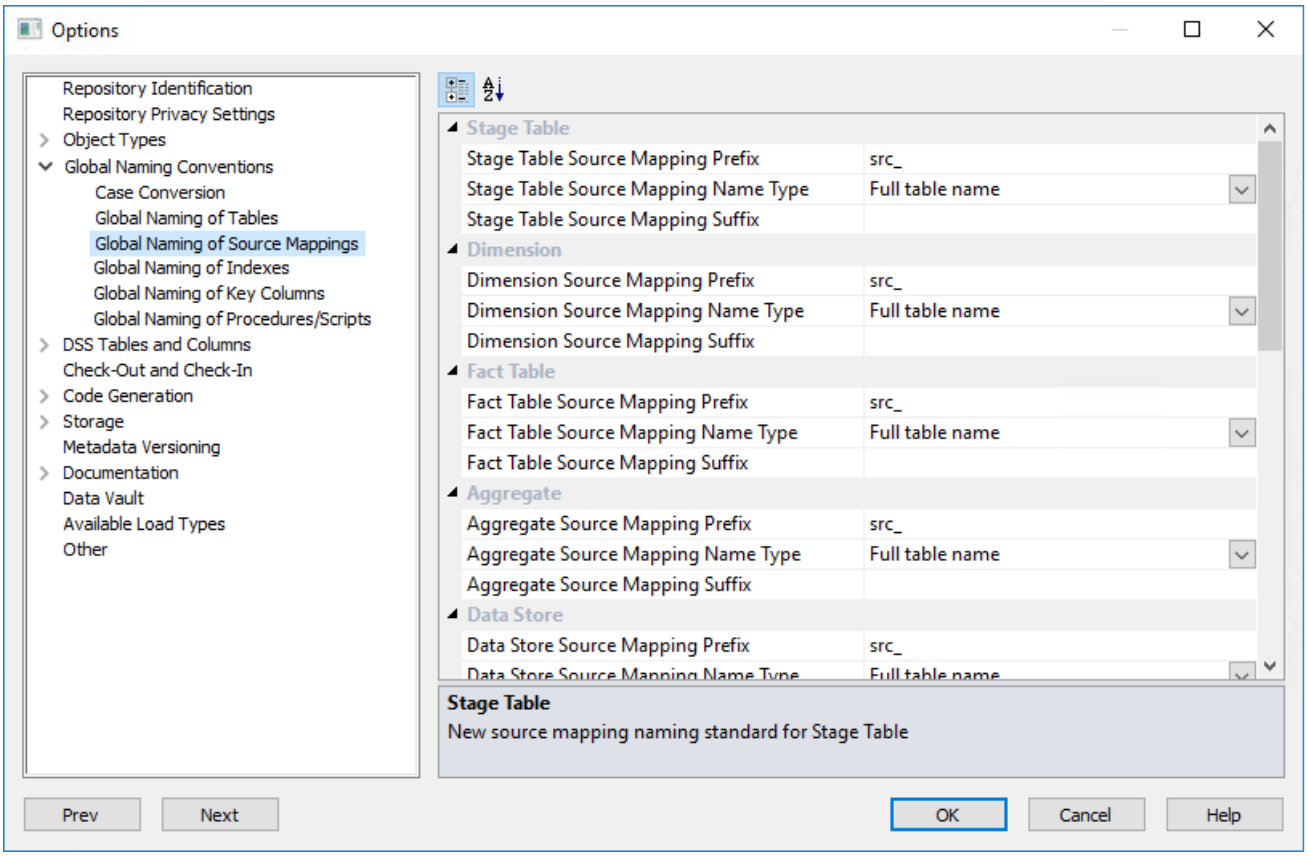


© Copyright, WhereScape Software Ltd All rights reserved.

Global Naming of Source Mappings

The **Global Naming Conventions** option includes the **Global Naming of Source Mappings** setting that enables you to define naming standards for the source mapping objects in RED.

Refer to **Global Naming Conventions > Global Naming of Source Mappings** for details.



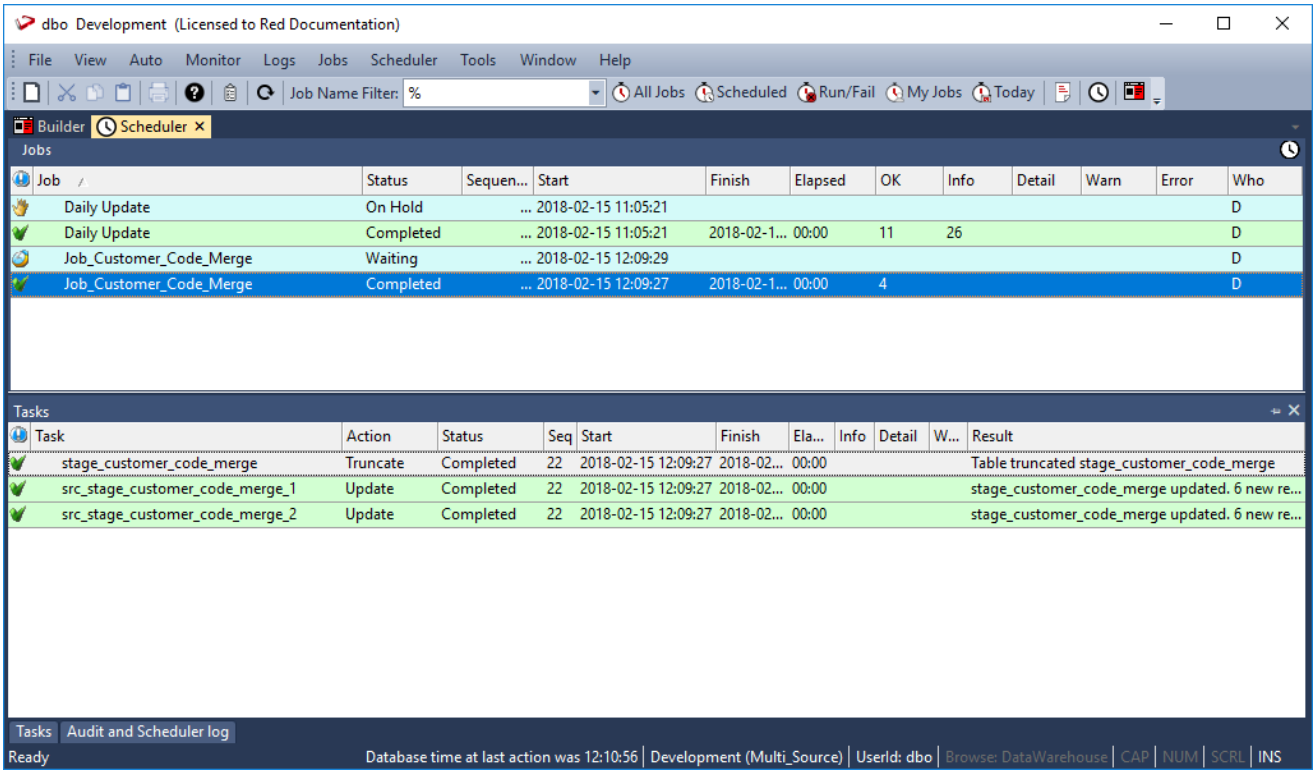
© Copyright, WhereScape Software Ltd All rights reserved.

Independent Execution of Update Procedures

The update procedures associated with the target table and the source mapping objects can be executed individually, separate from each other. Similar to other table objects in RED, the update procedures can be executed on demand or via the **Scheduler**. This provides the flexibility of scheduling the update procedures for each individual source mapping object as required, e.g. based on the update frequency of the tables where the source columns are obtained.

The update procedures for source mapping objects can only be generated using a template.

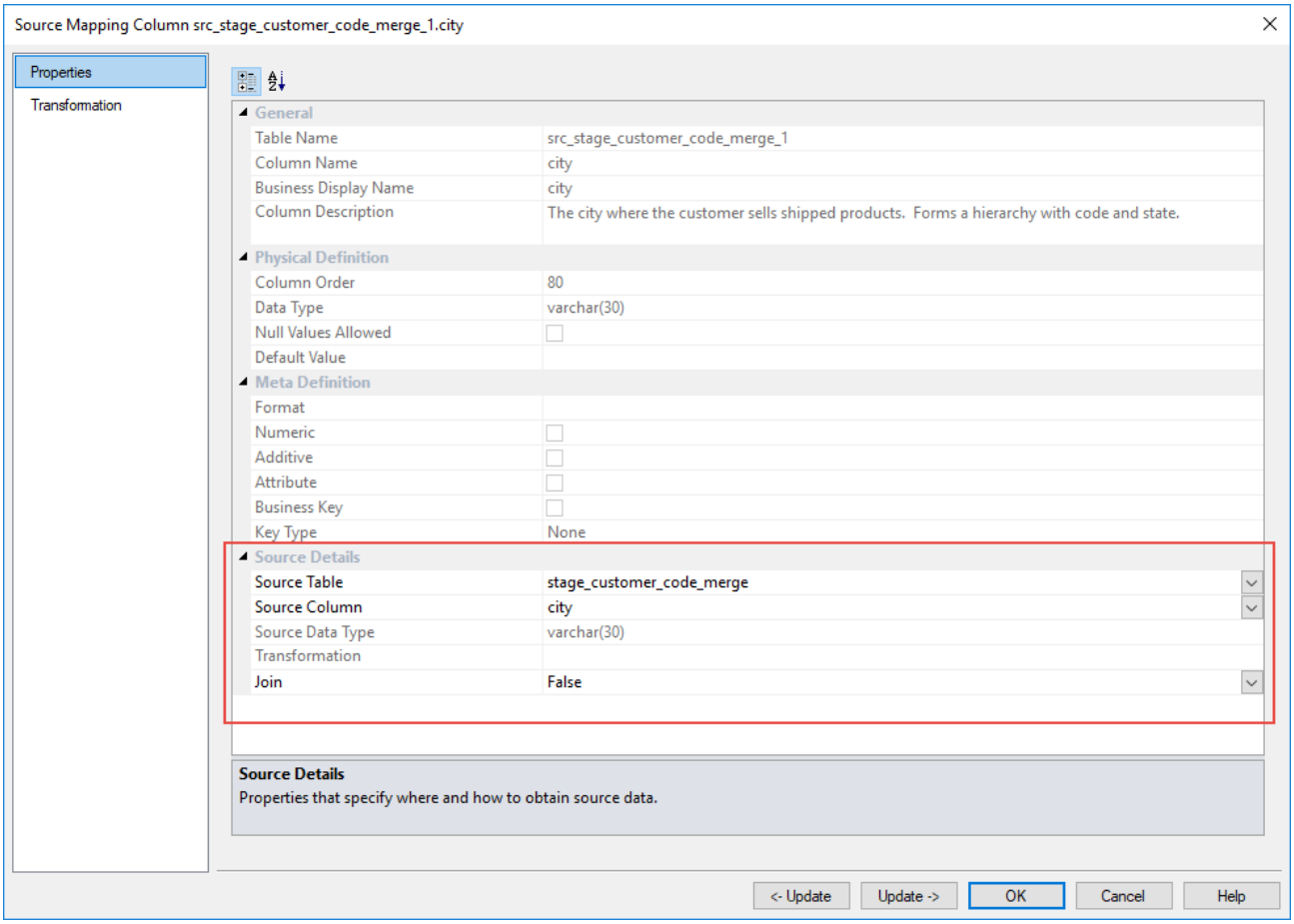
Refer to **Generating Update Procedures for Source Mapping Objects** for details



© Copyright, WhereScape Software Ltd All rights reserved.

Table Column Properties

The **Source Details** table column properties is used to specify where to obtain the source data and provides options for processing (transformation and join settings) the source data. You can also define and manage column source mappings from this screen.



Refer to the relevant table Column Properties sections in this user guide for details.

© Copyright, WhereScape Software Ltd All rights reserved.

Adding Source Mapping to Objects

When adding source columns to a table object in RED, the target table's current source columns are used to create the first source mapping object and then proceeds to create the second source mapping object which contains the additional source columns that need to be mapped to the target table. The columns from the source table that exists in the target table are automatically mapped.

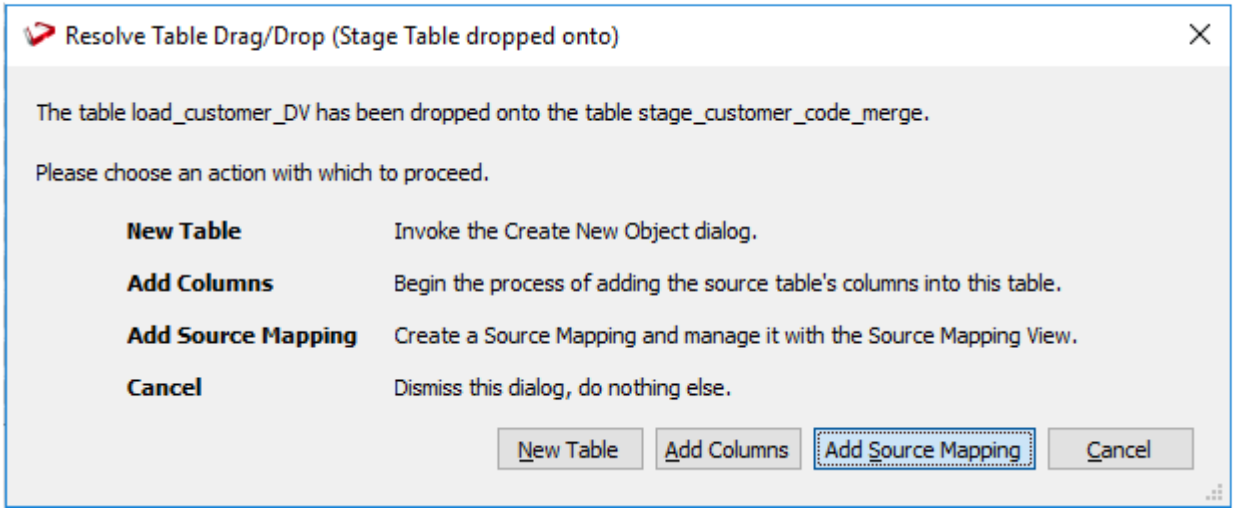
The process of adding Source Mappings is initiated when a source table is dragged and dropped to an existing table object in RED.

The user is prompted with the following options:

- New Table - Opens a new dialog that enables you to create the dropped table as a new object in the metadata repository.
- Add Columns - Inserts the columns of the dropped table into the target table.
- Add Source Mapping - Inserts a new source mapping into the target table.

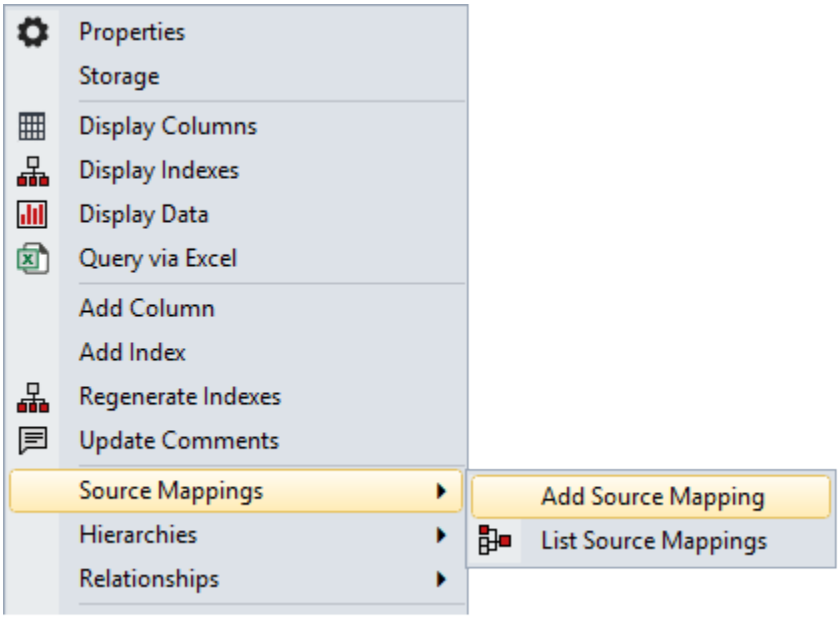
Note

If you are creating the first source mapping, the target table is first converted to a source mappable object, and then a new source mapping is inserted. If it is not the first source mapping, a new source mapping is just inserted into the target table.



© Copyright, WhereScape Software Ltd All rights reserved.

The alternative method is to right click the target table from the **Objects** list pane and then select the **Add Source Mapping** option from the context menu:



© Copyright, WhereScape Software Ltd All rights reserved.

Drag and Drop

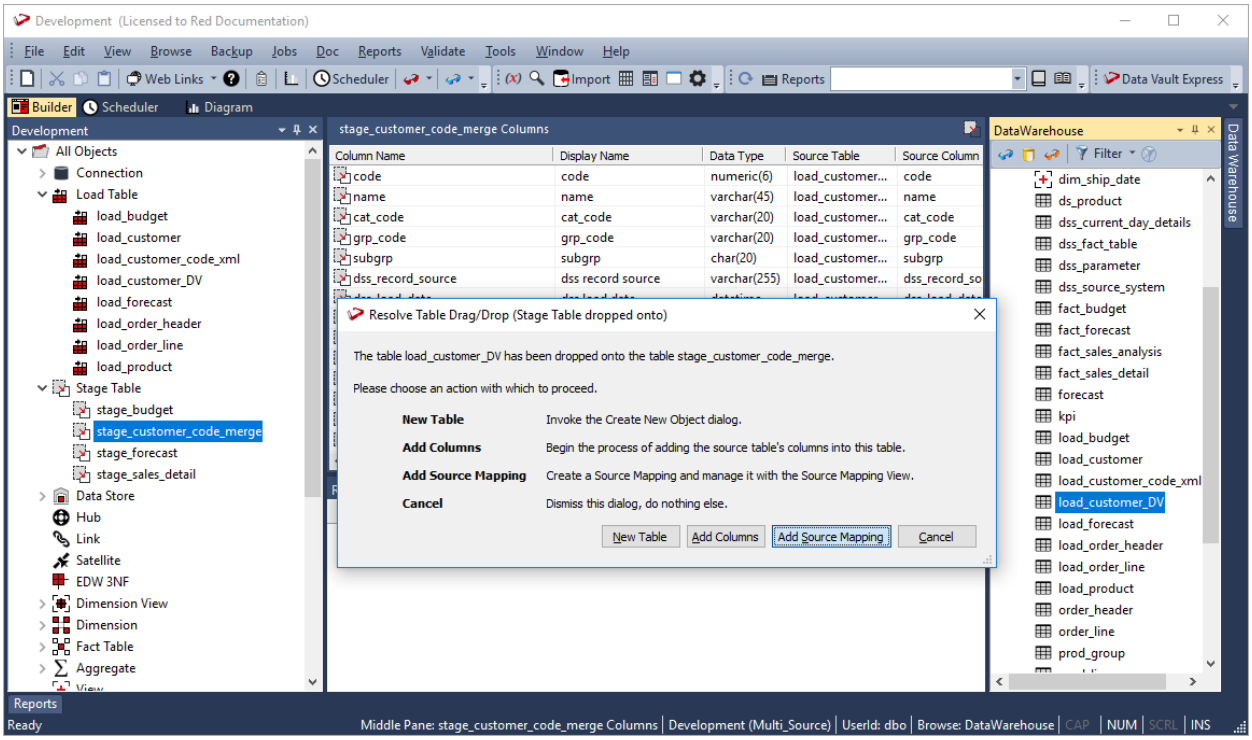
The common approach to create source mappings is to select the source table that contains the columns that you want to add and then drag this table into the target table.

This process creates two source mapping objects, the first one contains the original source mapping of the target table and second one is for the additional columns from the source table selected in the **Browser** pane.

You can drag additional source tables to create additional source mappings objects until you have all the source data required for your target table.

1. Click the target table from the **Objects** list in the left pane.

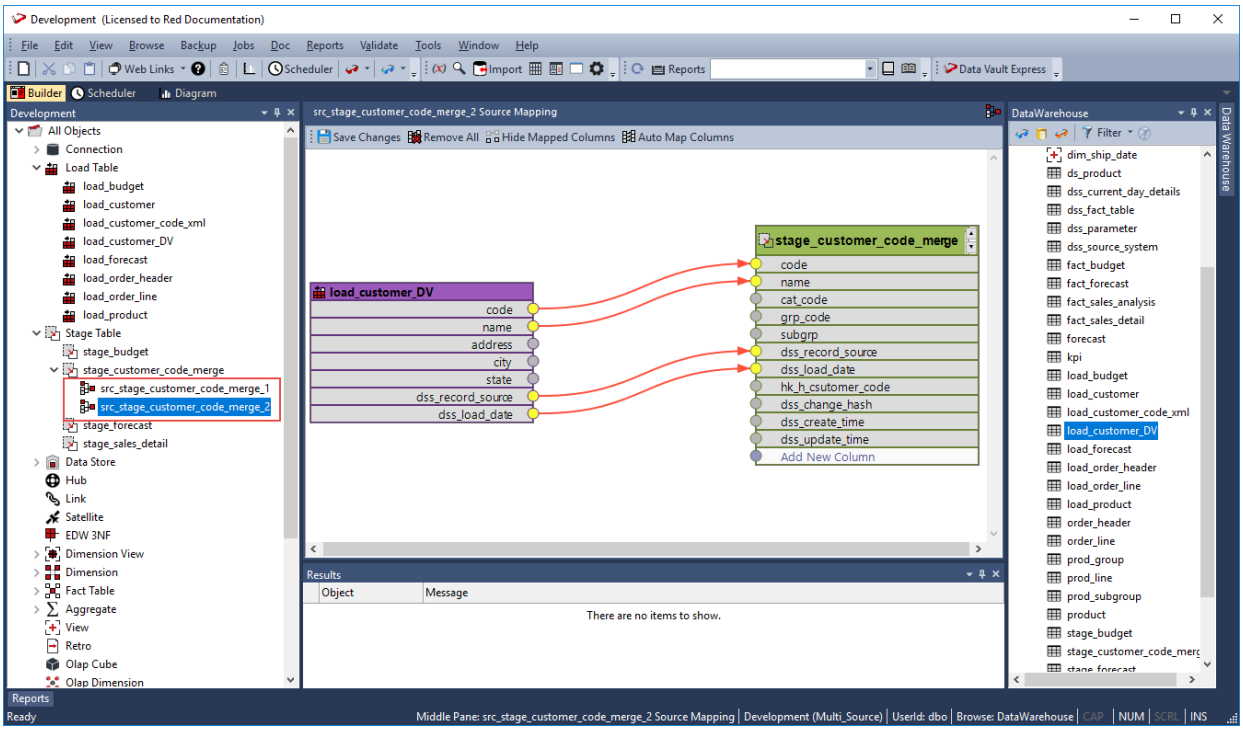
   The middle pane displays the properties of the selected target table, the pane is identified as a target for new Source Mapping objects.
2. Browse to the Data Warehouse via the **Browse > Source Table** menu option.
3. Click the source table that contains the columns you want to include as sources from the **Browser** pane, and drag it to the middle pane.
4. Click the **Add Source Mapping** button in the **Resolve Table Drag/Drop** prompt.



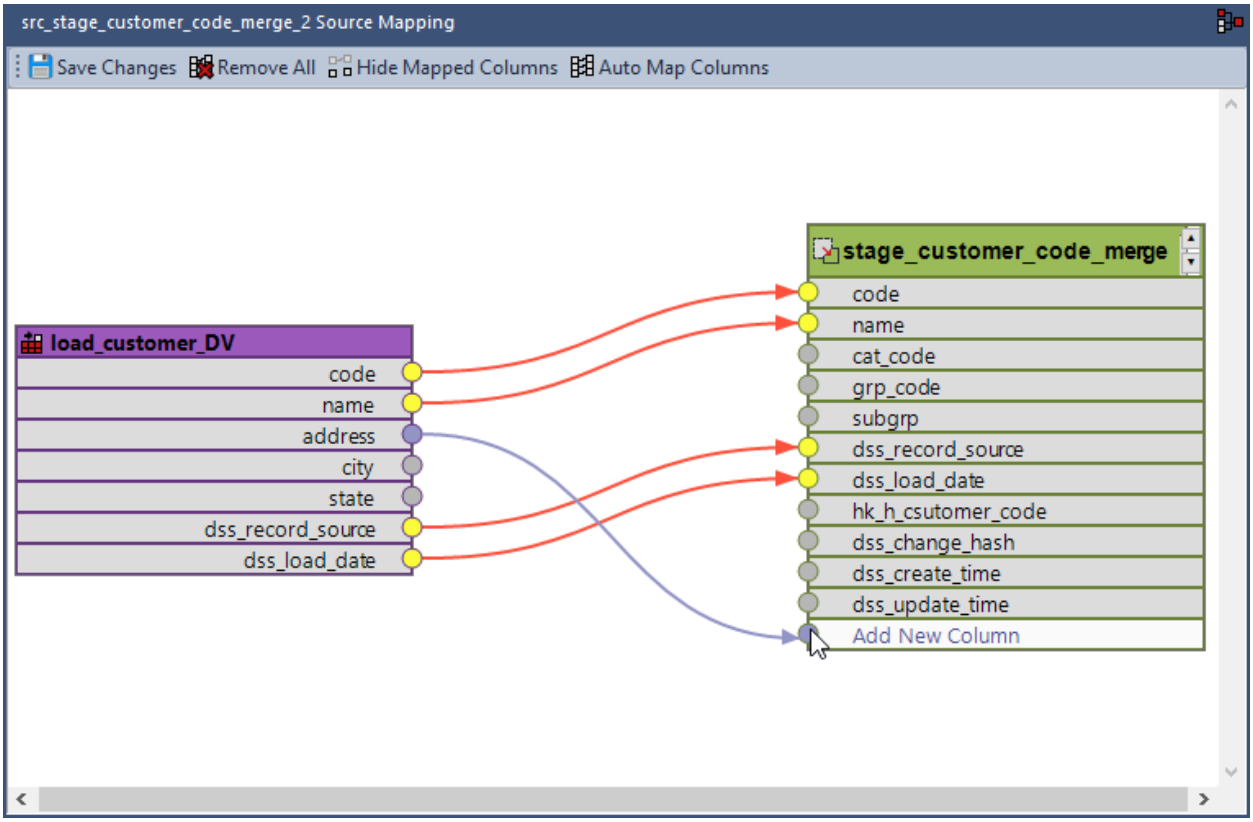
5. The new Source Mapping objects are created and displayed under the target table in the **Objects** list pane.

   The middle pane displays the Source Mapping tool which provides a graphical representation of the mapped columns between the second source table and the target table. The columns from the source table that exists in the target table are automatically mapped.

© Copyright, WhereScape Software Ltd All rights reserved.

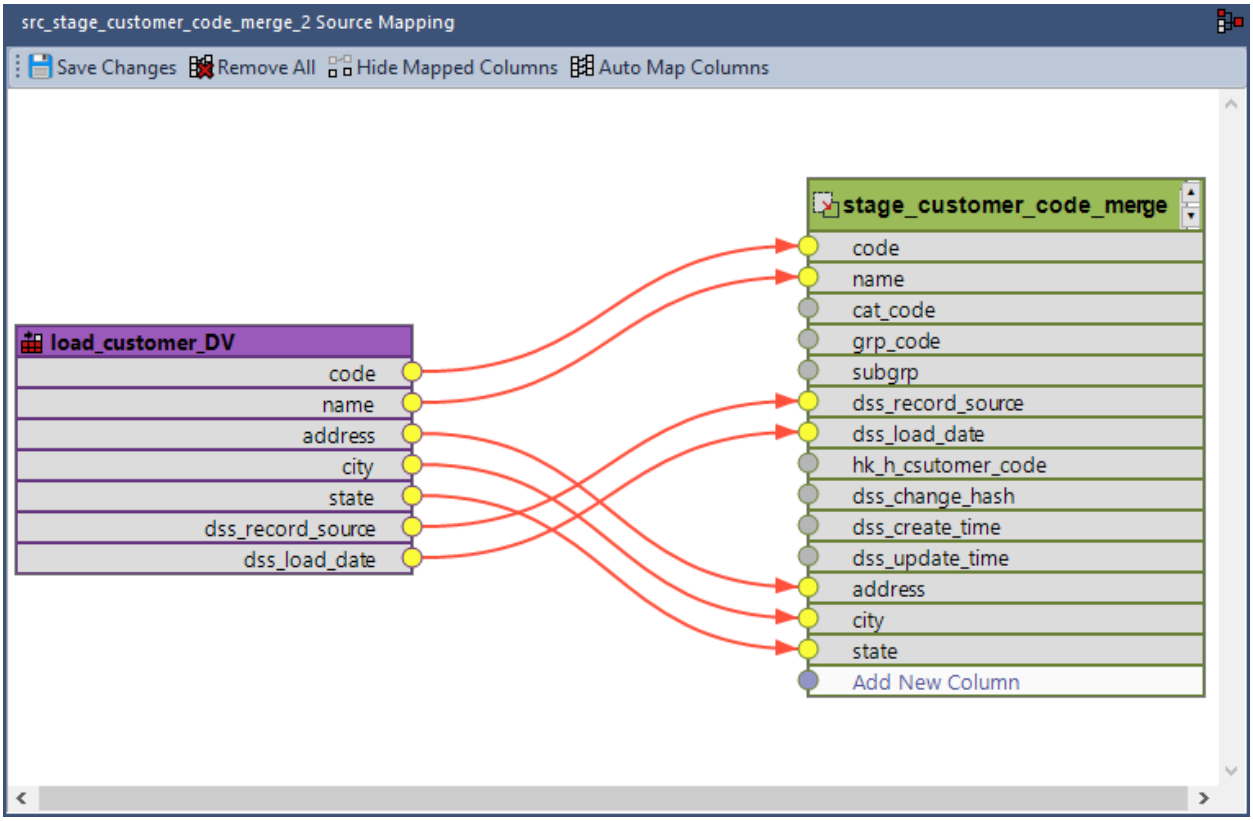


6. To map the additional source columns that does not exist in the target table, click the connection point of the source column you want to map and drag the line that appears to the connection point of the **Add New Column** row in the target table.

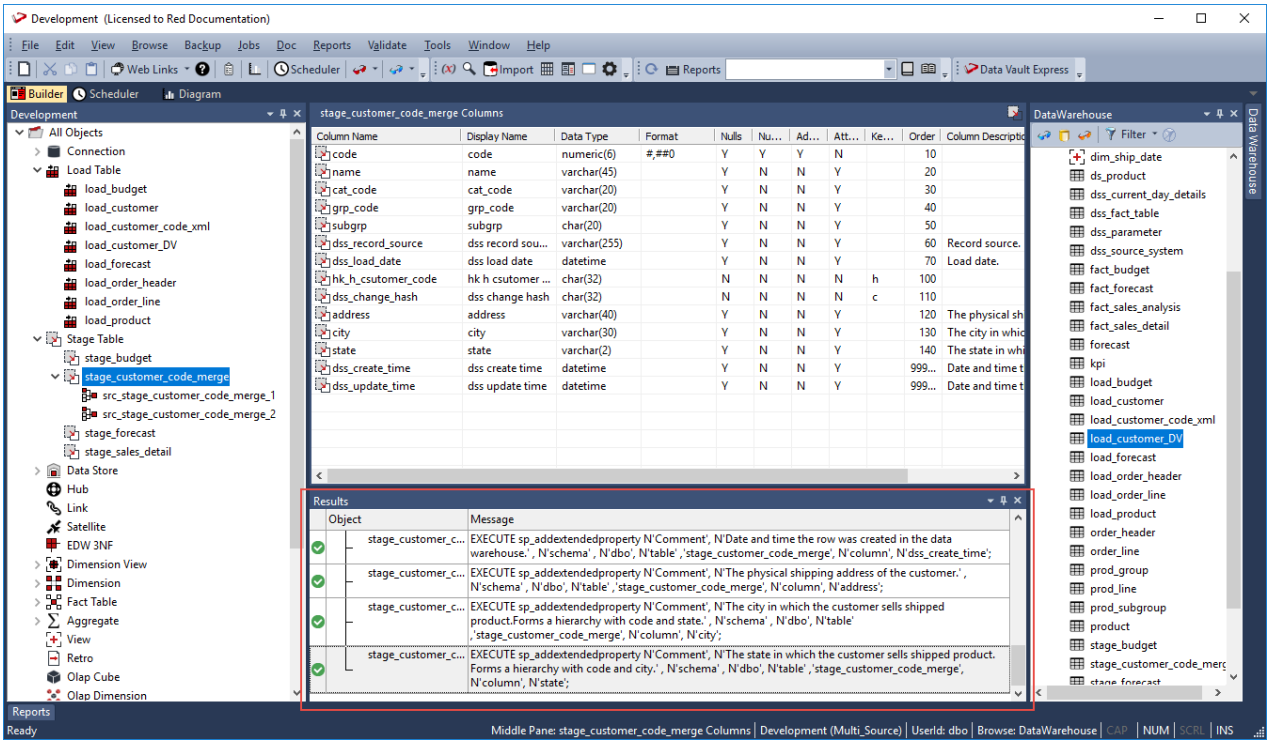


© Copyright, WhereScape Software Ltd All rights reserved.

7. Repeat the same steps to map all the other source columns to the target table.



8. Click **Save Changes** to save the defined source mappings.
9. Right-click the target table from the **Objects** list in the left pane and select **Create(ReCreate)** from the context menu to recreate the table. The **Results** pane displays confirmation that the target table was successfully recreated.



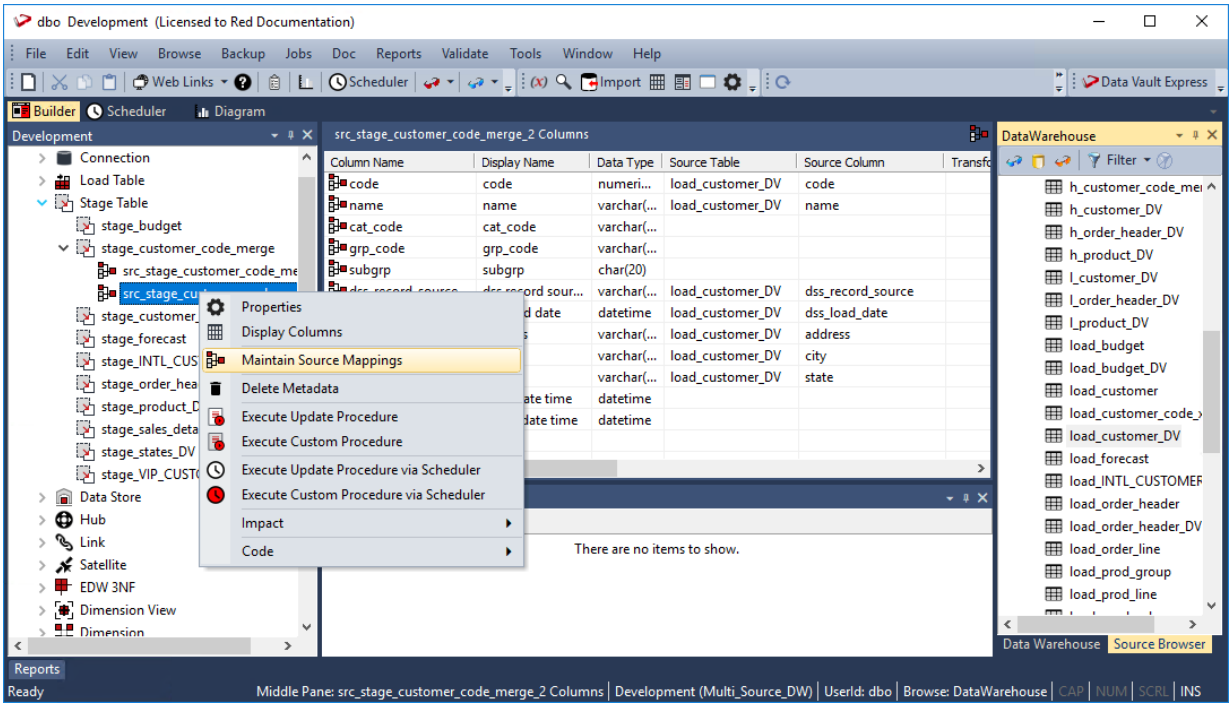
Once the target table is defined and created, you need to generate the update procedure for each source mapping object to populate the target table. The update procedure is created using a template—refer to **Generating Update Procedures for Source Mapping Objects** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

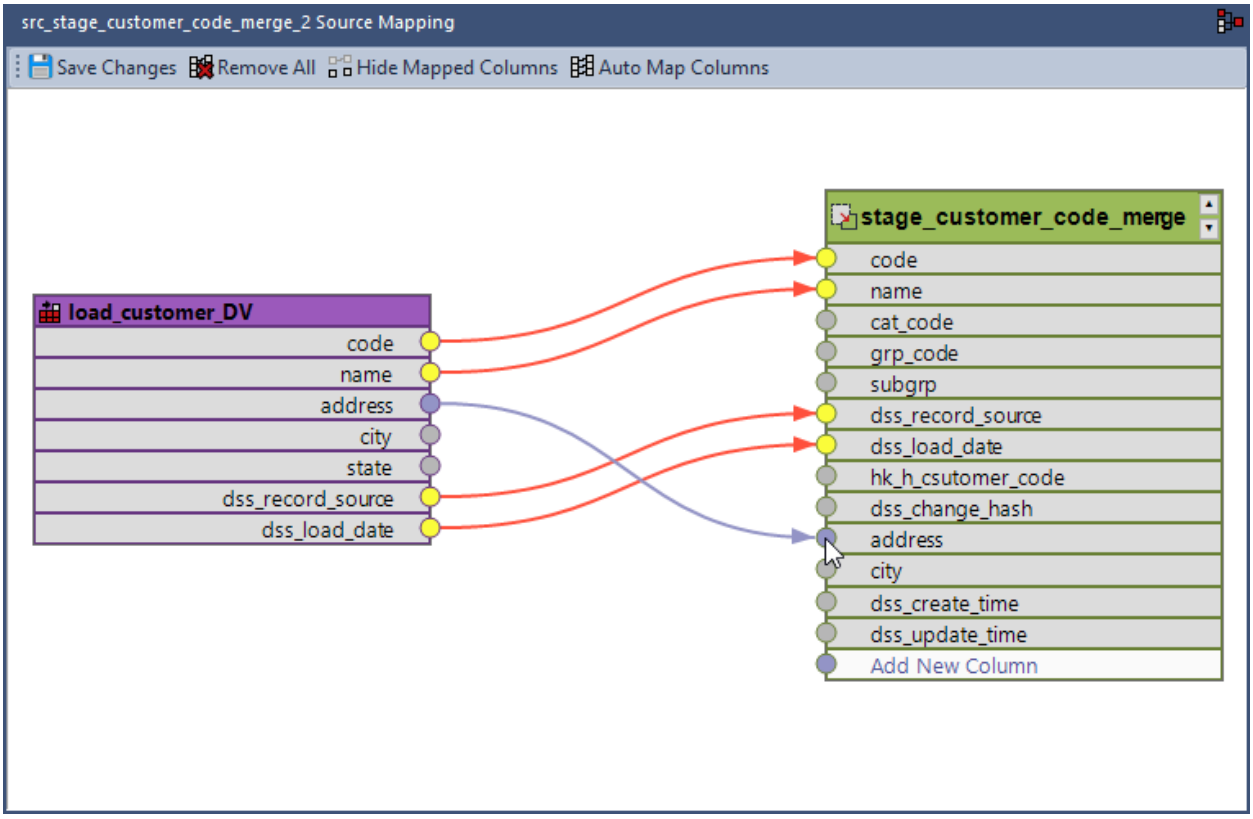
Maintaining Source Column Mappings

The following describe the process for managing source mappings between the source table(s) and the target table.

1. Right click the Source Mapping object from the left-pane and then select **Maintain Source Mappings** from the context menu.



2. Click the connection point of the source column you want to map and then drag the line that appears to the connection point of the target column.

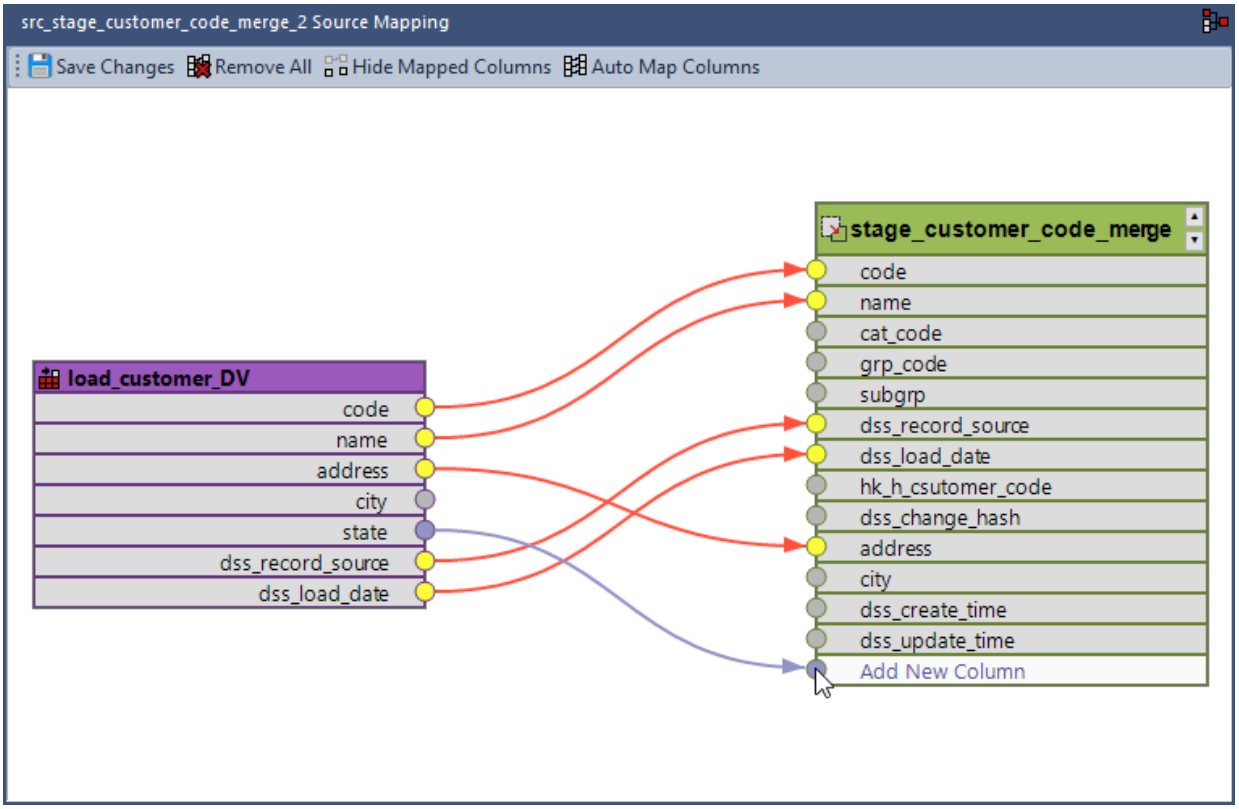


© Copyright, WhereScape Software Ltd All rights reserved.

Tips:

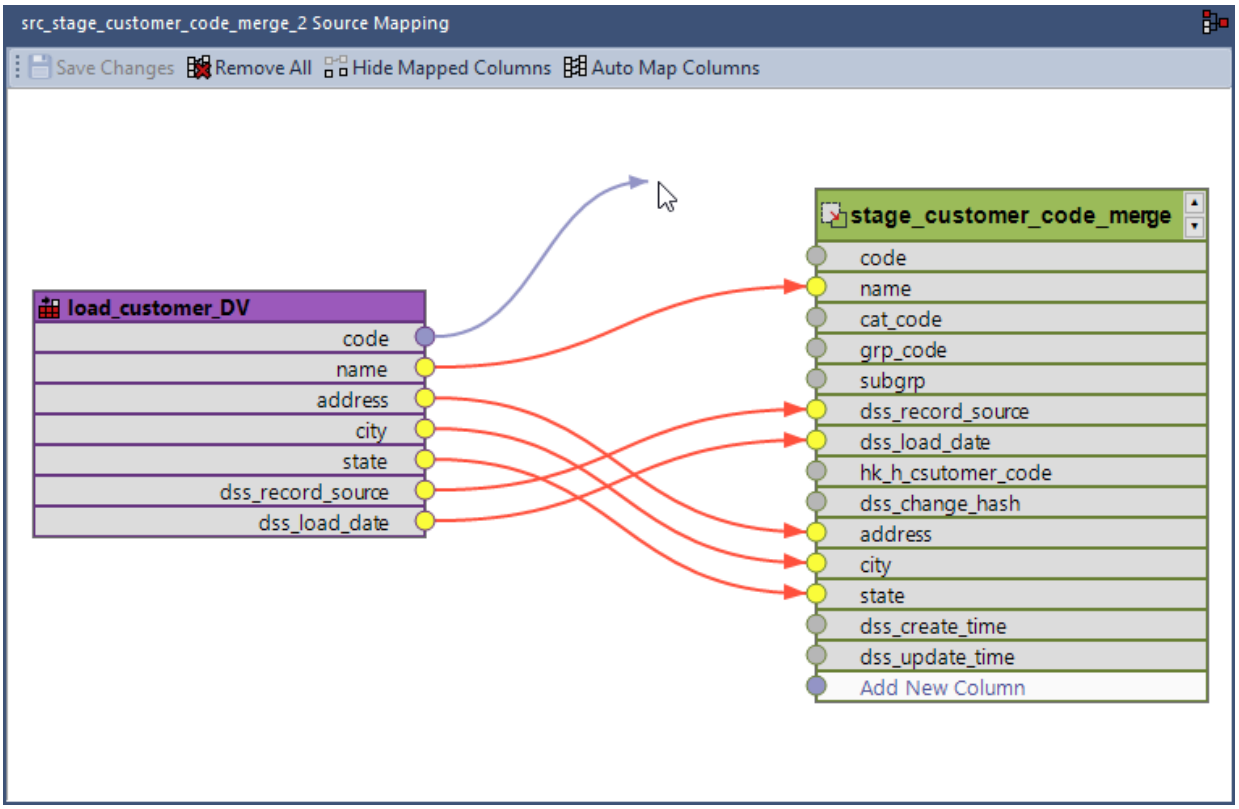
- The target table can be clicked and dragged vertically (up/down) within the diagram view. This is useful when there are more columns in the target table that can fit in the visible area of the diagram view. You can also use the mouse wheel for scrolling up/down the diagram view.
- You can undock any of the panes (**Object**, **Browser** and **Results** pane) in the **Builder window**, to gain a bigger work area for the Source Mapping tool.

3. To map a source column that does not exist in the target table, click the connection point of the source column you want to map and then drag the line that appears to the connection point of the **Add New Column** row in the target table.



4. To remove an existing source mapping, click the connection point from the target table and then drag away the line that appears to disconnect.

© Copyright, WhereScape Software Ltd All rights reserved.



Notes:

- Clicking **Remove All** removes all the current mappings between the source table and the target table.
- Clicking **Hide Mapped Columns** hides all the current mappings and only the unmapped columns are displayed. This is useful when there are more columns in the target table that can fit in the visible area of the diagram view and you only want to work on the columns that are not mapped.
- Clicking **Auto Map Columns** automatically maps all columns from the source table that exists in the target table. This is useful if you removed all or some of the mappings and would like to restore them.

© Copyright, WhereScape Software Ltd All rights reserved.

Generating Update Procedures for Source Mapping Objects

After successfully defining and creating the Source Mapping objects, you can generate the update procedure for each object via a template to populate the target table.

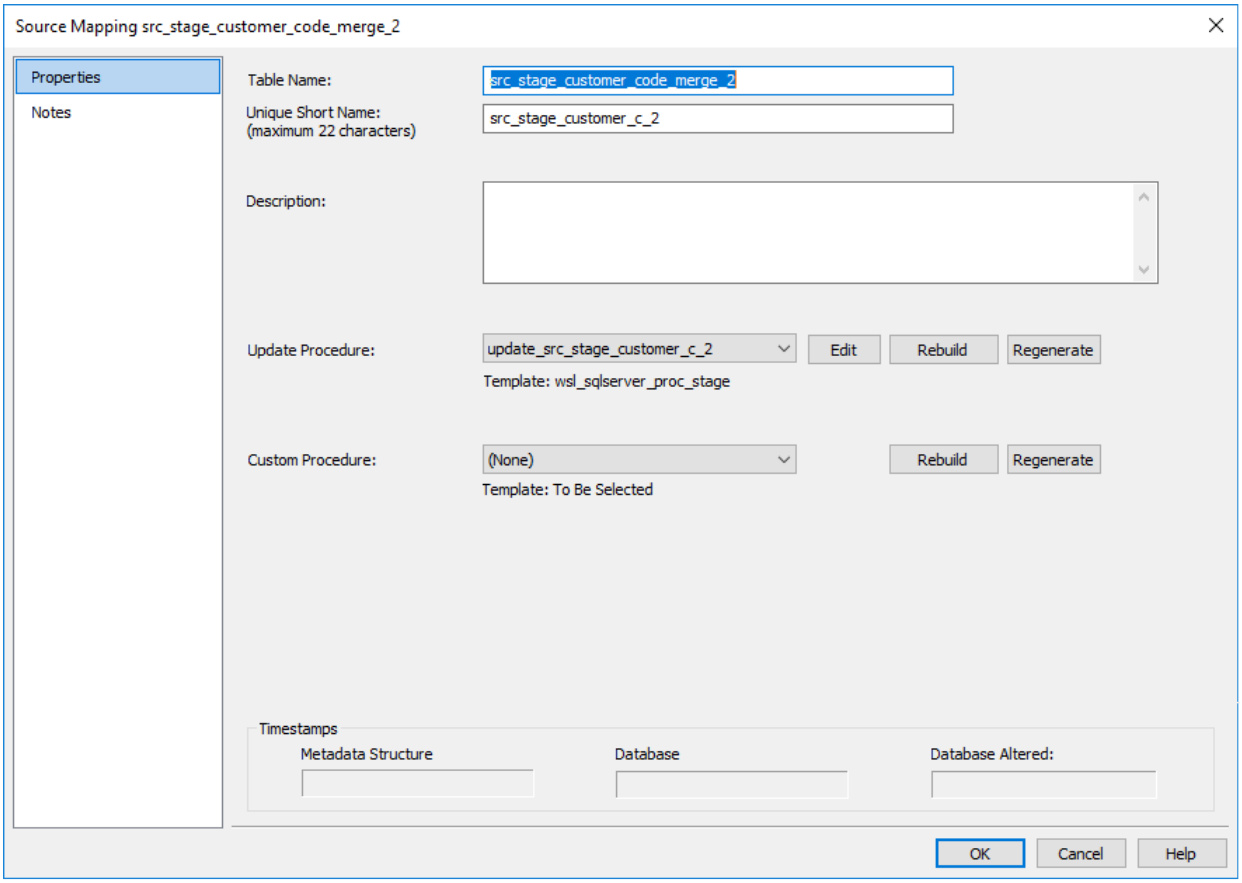
Generating update and custom procedures for Source Mapping objects is completed using the same workflow as all objects in RED that support template based procedure/script generation.

Executing a procedure associated with the Source Mapping object can be performed both interactively from the object's right-click context menu or via the Scheduler.

Notes:

- Please ensure that you have created your own Source Mapping object templates, before performing the steps below.
- You can also have an update or custom procedure defined on the target table, if a single consolidated procedure to process the table is required.

1. Click the **Rebuild** button beside the **Update Procedure** drop-down in the **Properties** screen of the Source Mapping object, to launch the procedure generation Wizard.

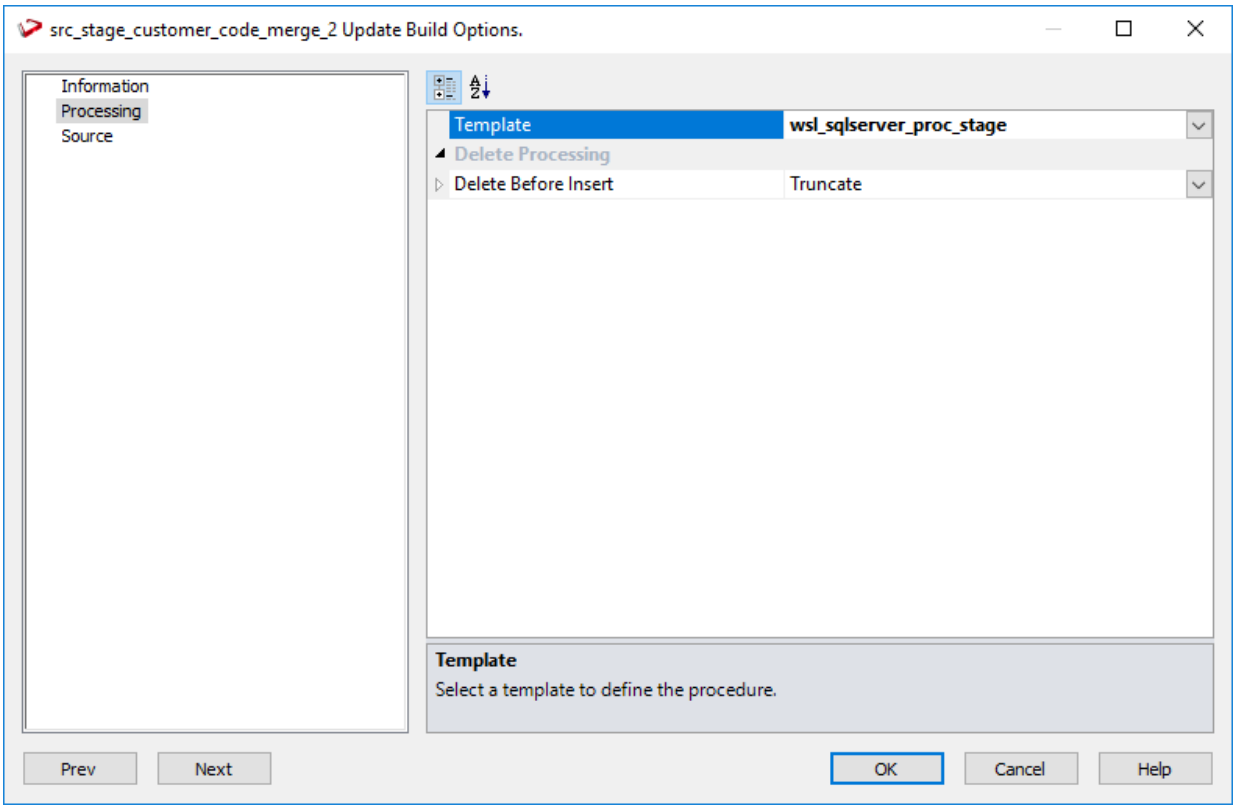


Note:

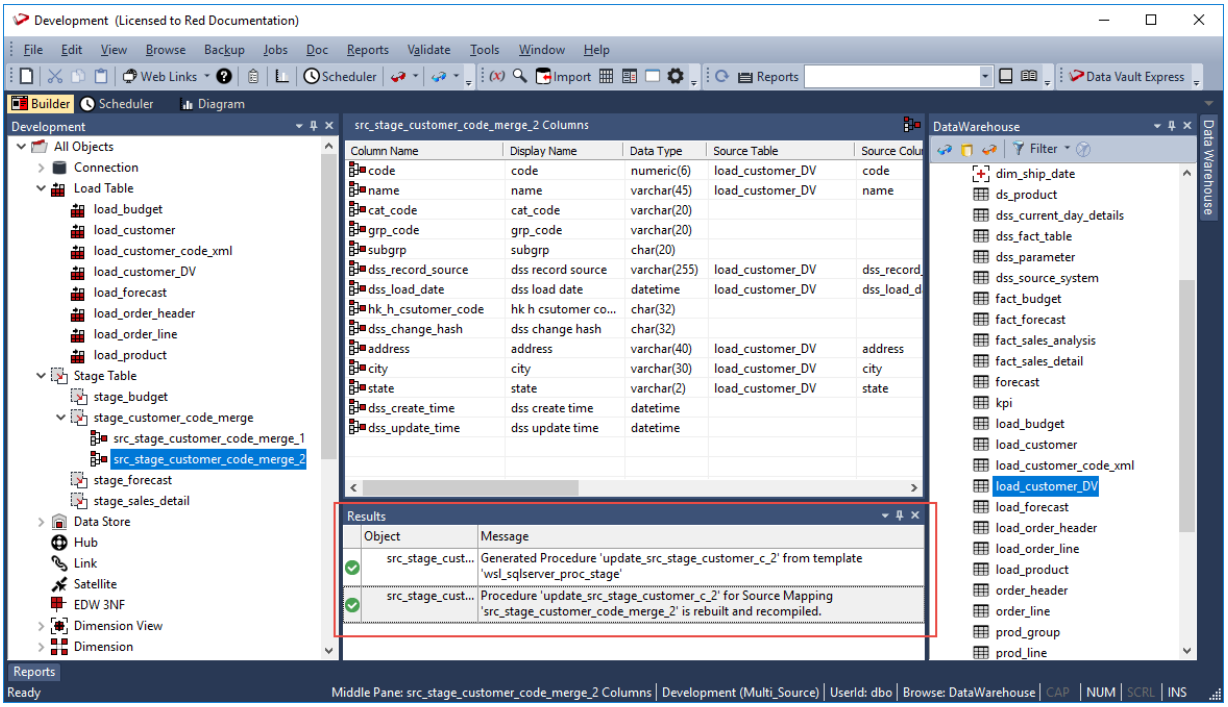
RED displays the name of the previously used update/custom procedure template below the **Update Procedure** and **Custom Procedure** drop-down fields by default.

© Copyright, WhereScape Software Ltd All rights reserved.

2. On the **Processing** tab of the **Update Build Options** screen, select the template to use from the **Template** drop-down or use the previous update procedure template. Configure the other options available as required.

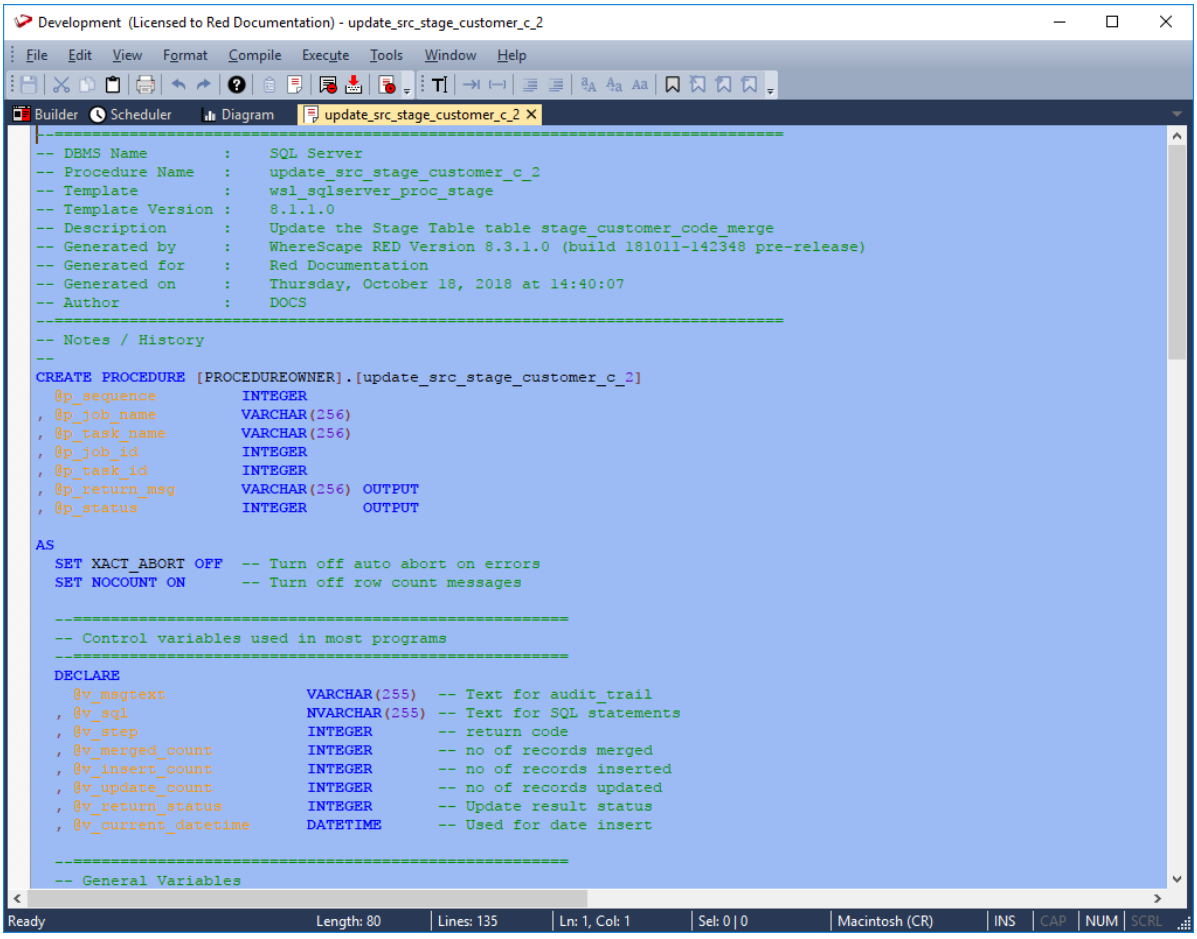


3. Click **OK** to proceed with the procedure generation. The **Results** pane displays confirmation that the procedure was generated.

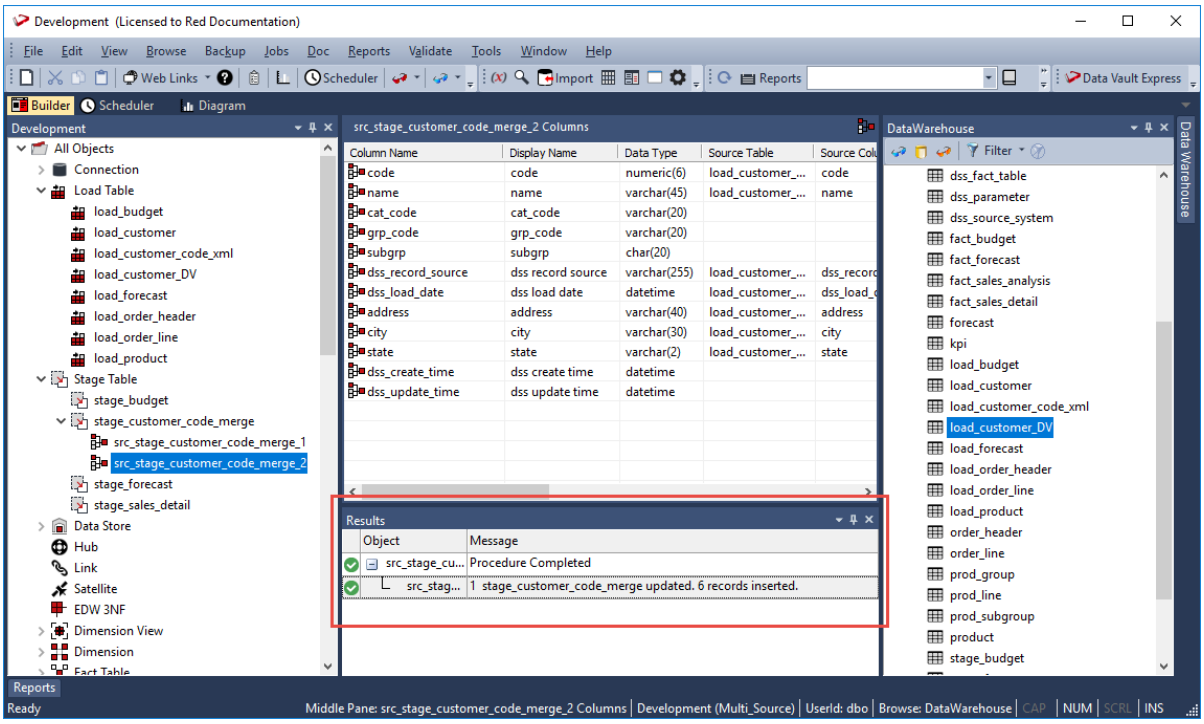


© Copyright, WhereScape Software Ltd All rights reserved.

4. Right-click the source table from the left pane, and select **Code > View update** from the context menu to view the contents of the update procedure generated.



5. Right-click the source table from the left pane and select **Execute Update Procedure** from the context menu to run the procedure. The **Results** pane displays the number of records created.



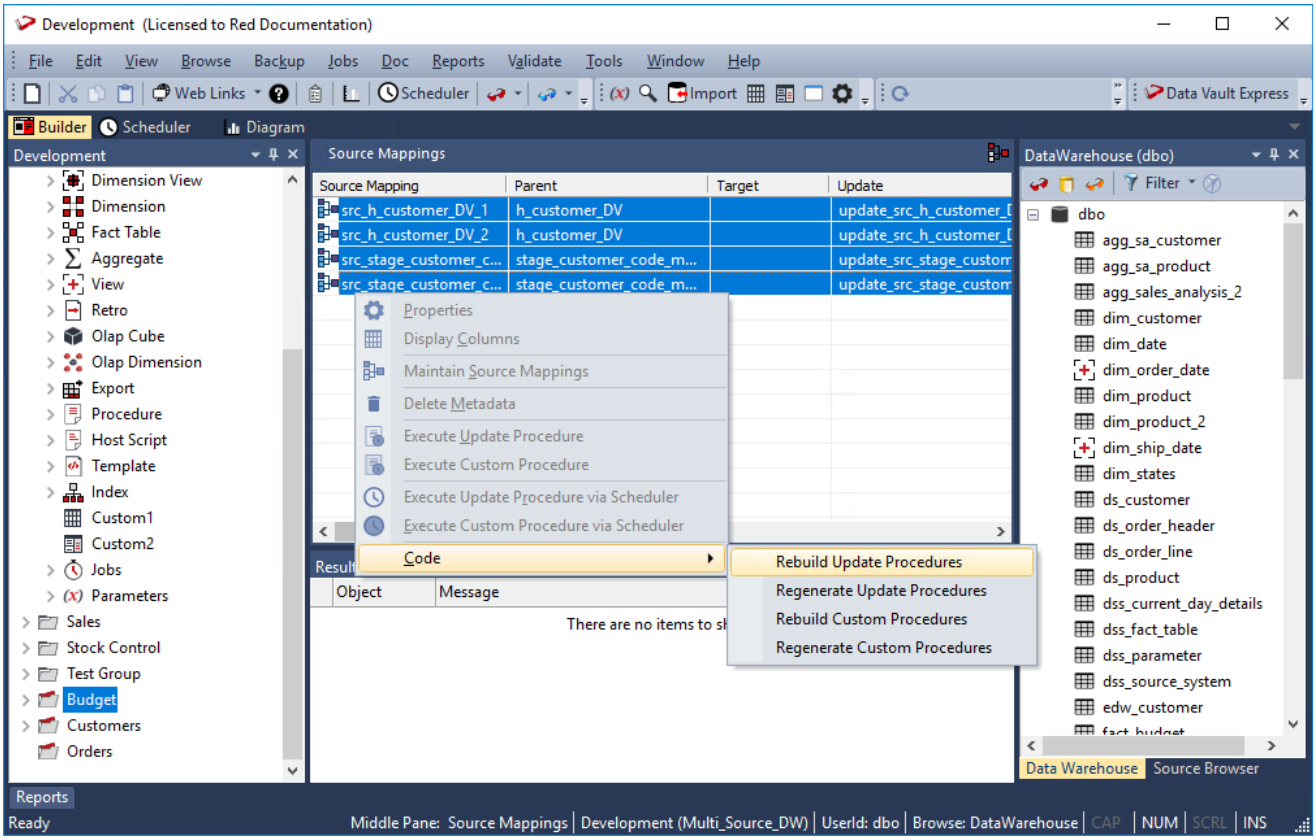
Repeat the same steps described above to create update procedures for the other source mapping objects, defined for the target table.

© Copyright, WhereScape Software Ltd All rights reserved.

Bulk Update Procedure Regeneration

Once the update procedure code has been generated for the source mapping objects, RED enables you to perform bulk code regeneration in source mapping objects from multiple tables.

You can list all the source mapping objects in a Project, Project Group or Object Group and then perform bulk update via the right click context menu:



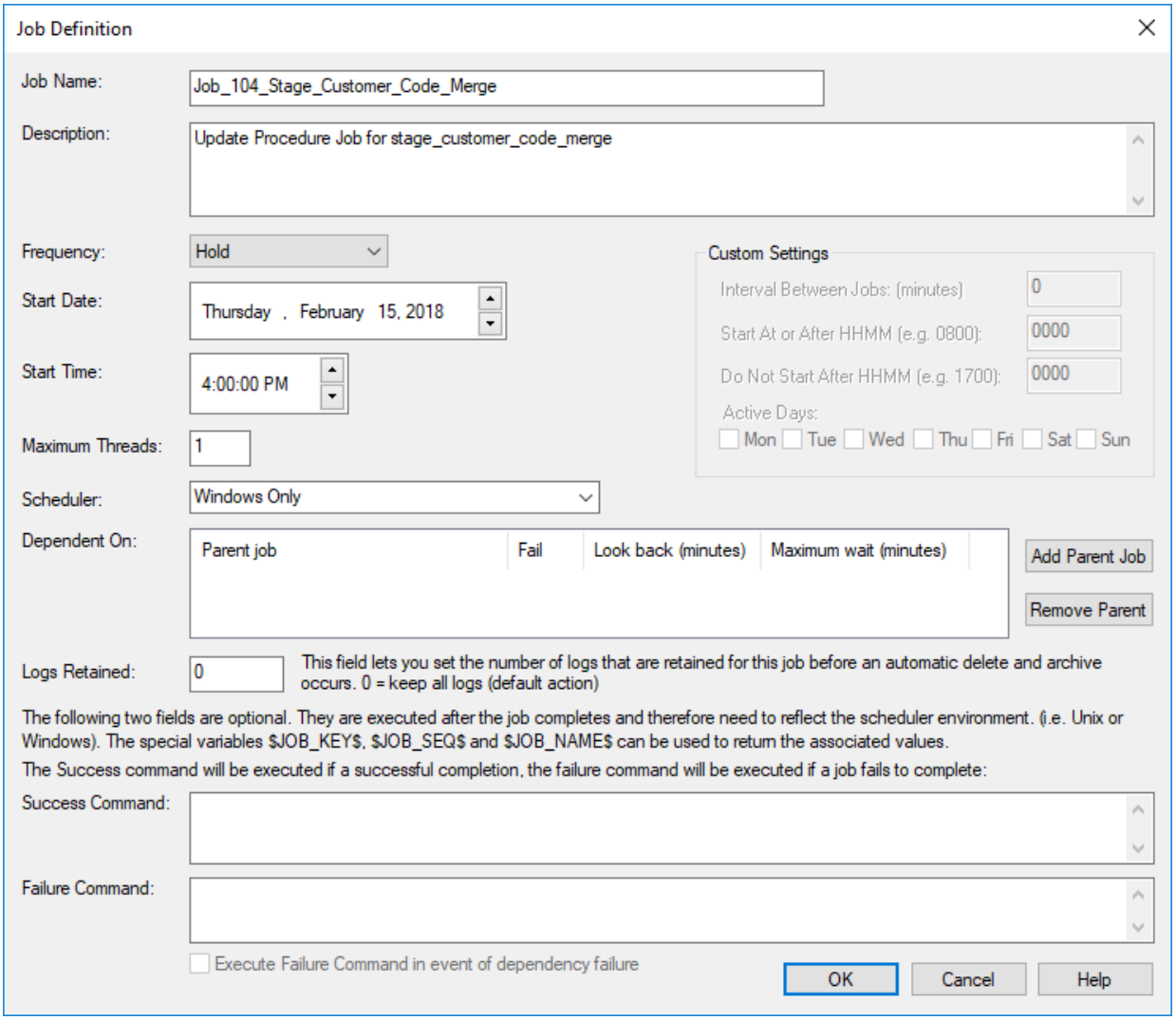
Refer to **Listing Source Mapping Objects** for details

© Copyright, WhereScape Software Ltd All rights reserved.

Executing Update Procedures via Scheduler

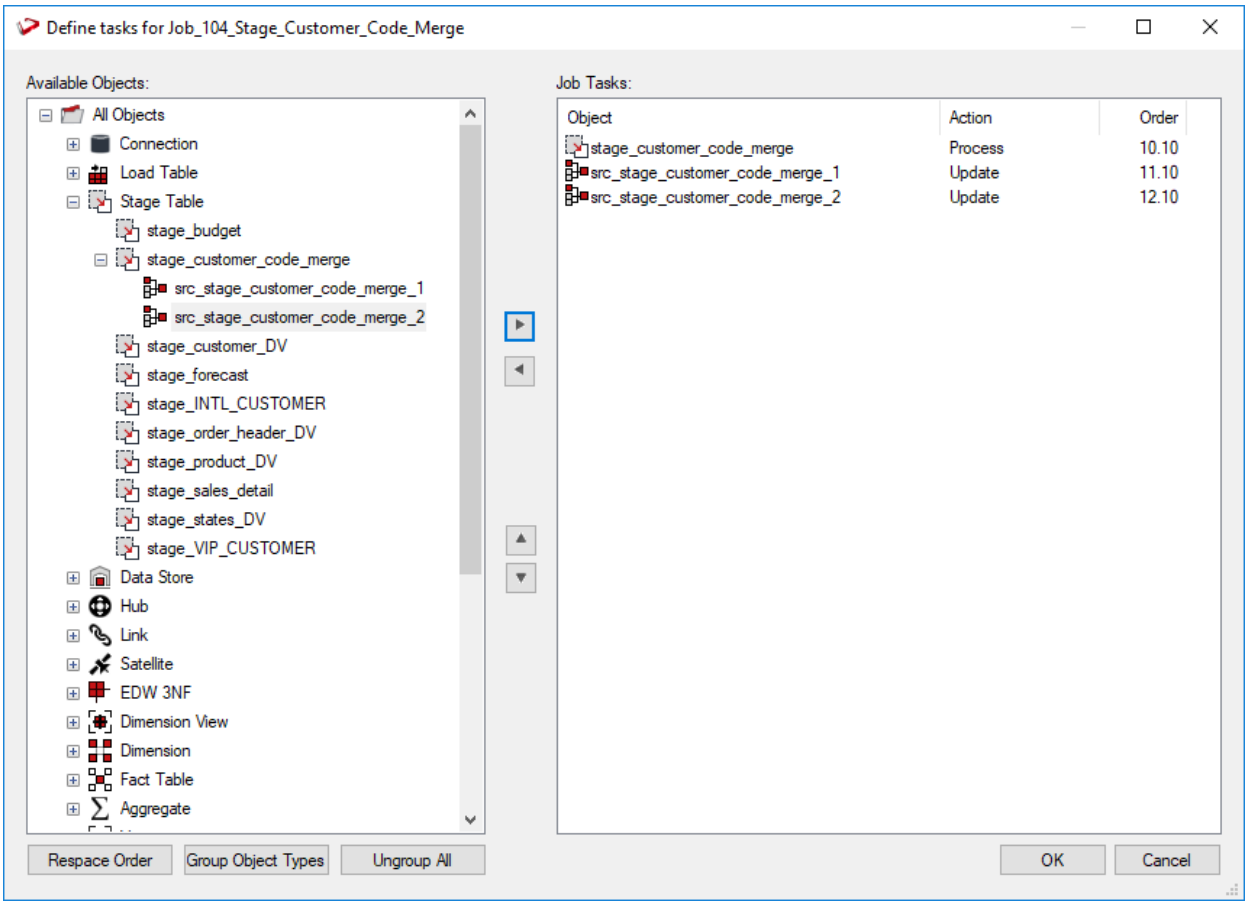
The update procedures associated with the target table and the source mapping objects can be executed individually, separate from each other. Similar to other table objects in RED, the update procedures can be executed via the Scheduler.

1. Click the **Scheduler** tab to open the **Scheduler** window.
2. Click the **New Job** button from the **Scheduler** window toolbar to create a new job
3. Complete the fields in the **Job Definition** window. Refer to **Creating a Job** for the descriptions of the fields.



4. On the **Define tasks** window, select the target object and the its associated source mapping objects from the **Available Objects** pane and add them to **Job Tasks** pane to define the tasks to be run for the job.

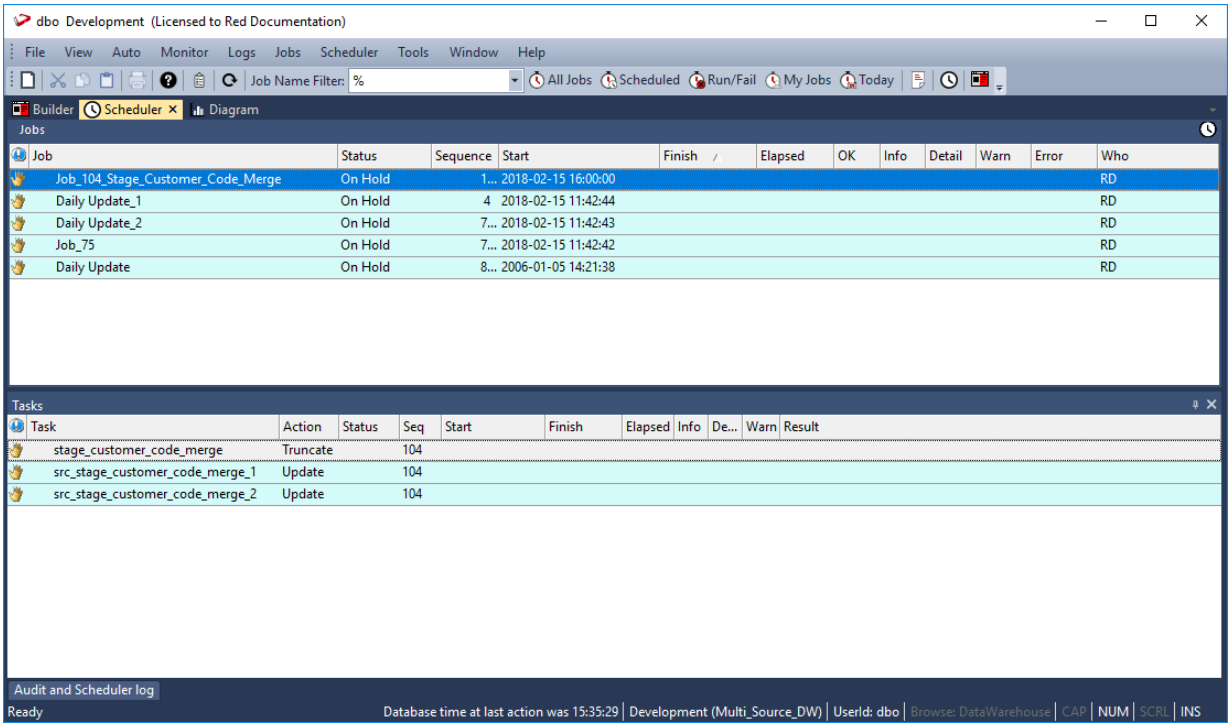
© Copyright, WhereScape Software Ltd All rights reserved.



Note:

The update procedures associated with the target table and the source mapping objects can be executed individually, separate from each other.

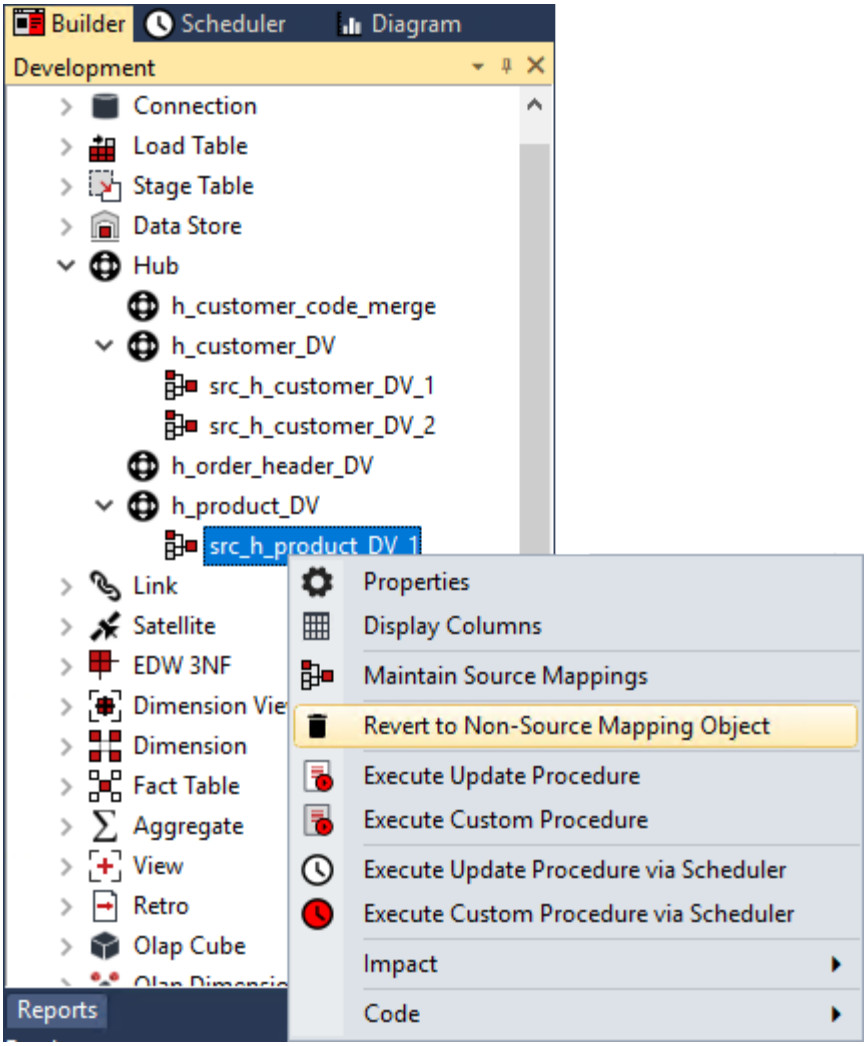
5. Click **OK** to save and exit to the **Scheduler** window.



© Copyright, WhereScape Software Ltd All rights reserved.

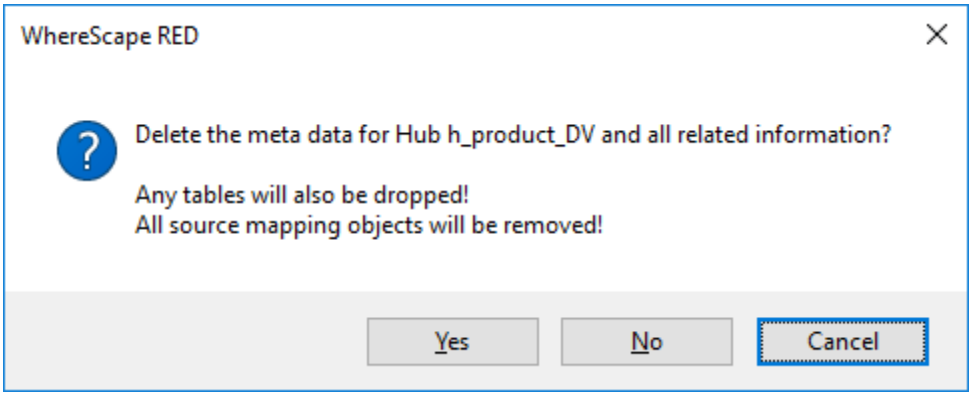
Reverting to Non Source Mapping Object

You can revert the target table to a non-source mapping object if you delete its associated source mapping objects. If only a single source mapping object is left, right clicking the remaining source mapping object provides the option **Revert to Non-Source Mapping Object**.



Note

Deleting the target table, deletes all its source mapping objects, including its associated meta data.



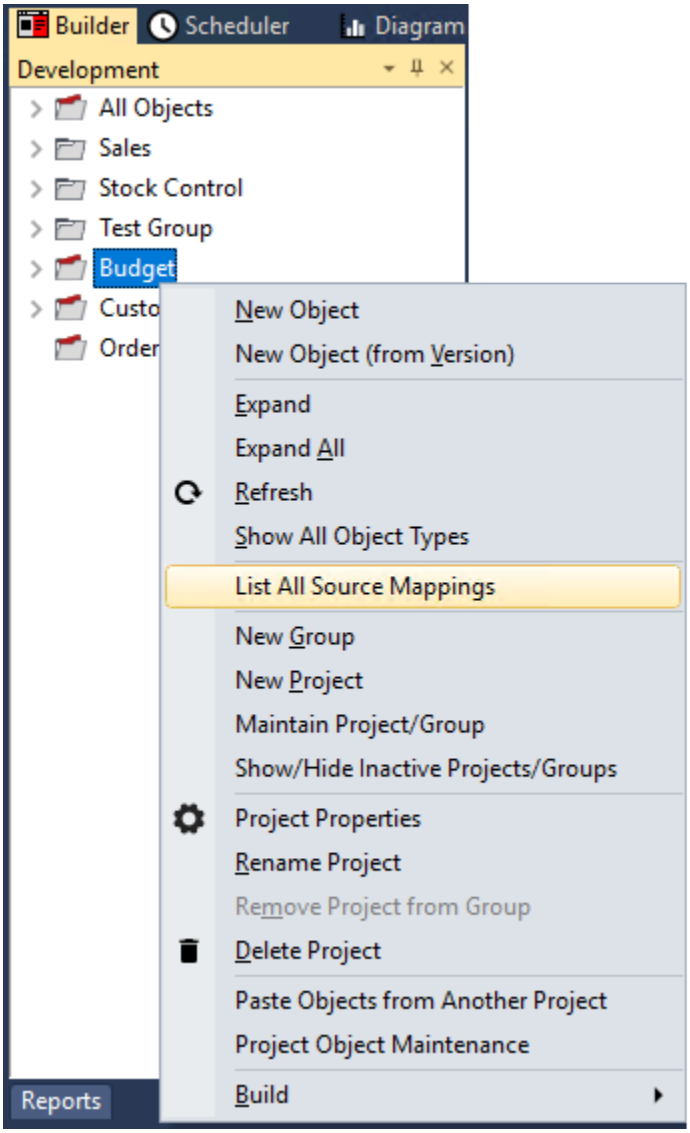
© Copyright, WhereScape Software Ltd All rights reserved.

Listing Source Mapping Objects

You can view a list of source mapping objects that are defined in a **Project, Project Group or Object Group** in RED. This enables you to quickly view and manage source mapping objects that exist within your data warehouse.

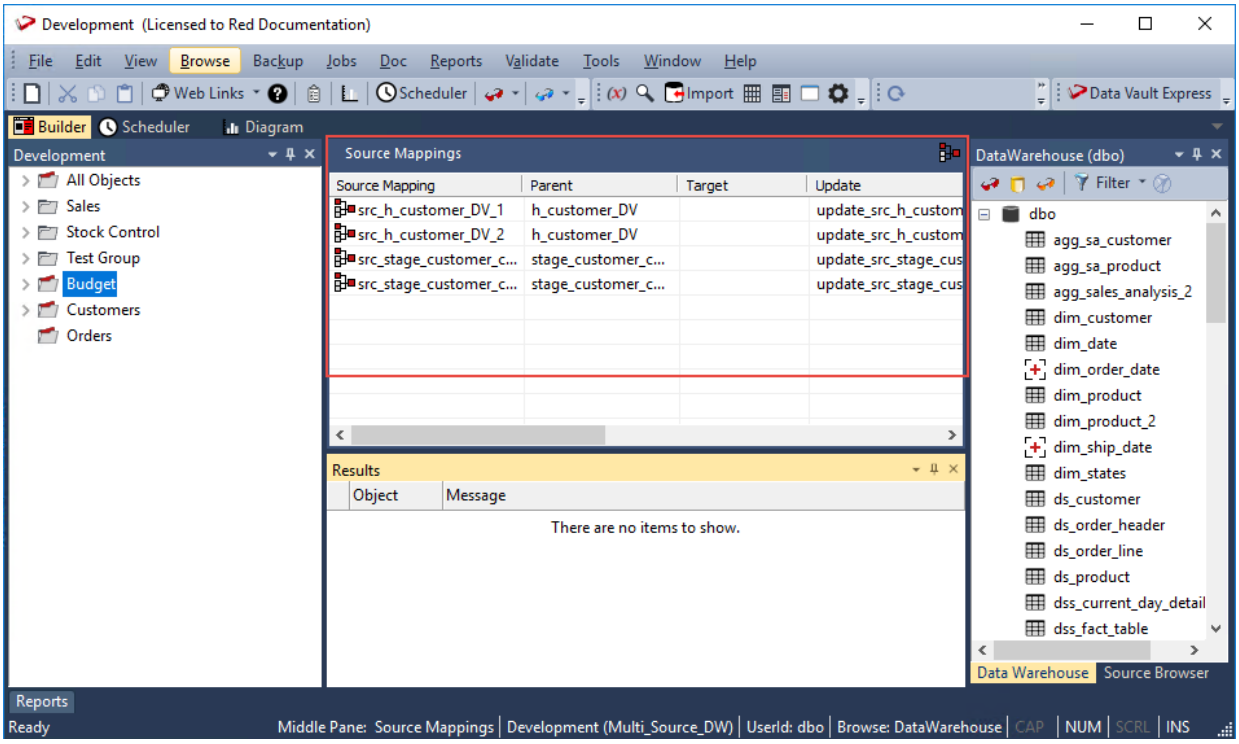
List Source Mapping Objects in a Project or Project Group:

1. Right-click the **Project** or **Project Group** from the left pane and select **List All Source Mappings** from the context menu:



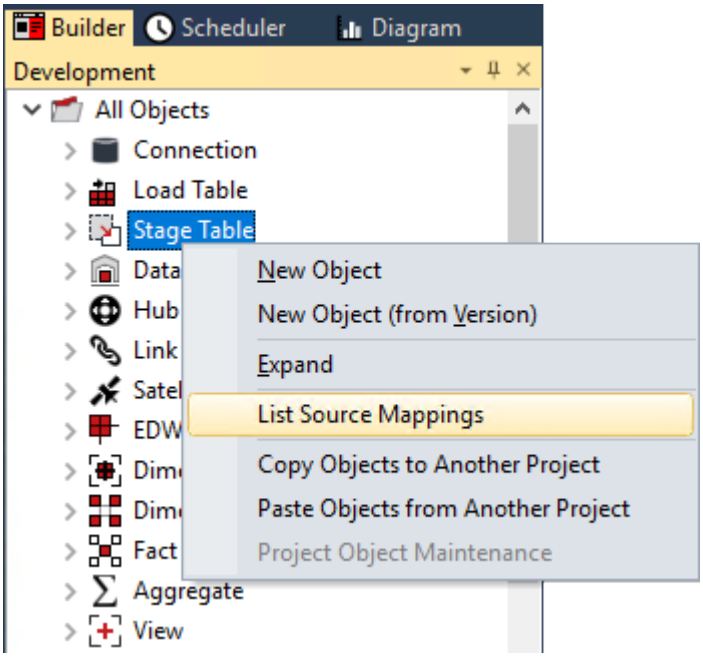
© Copyright, WhereScape Software Ltd All rights reserved.

2. The middle pane displays a list of all the source mapping objects in the selected Project or Project Group:



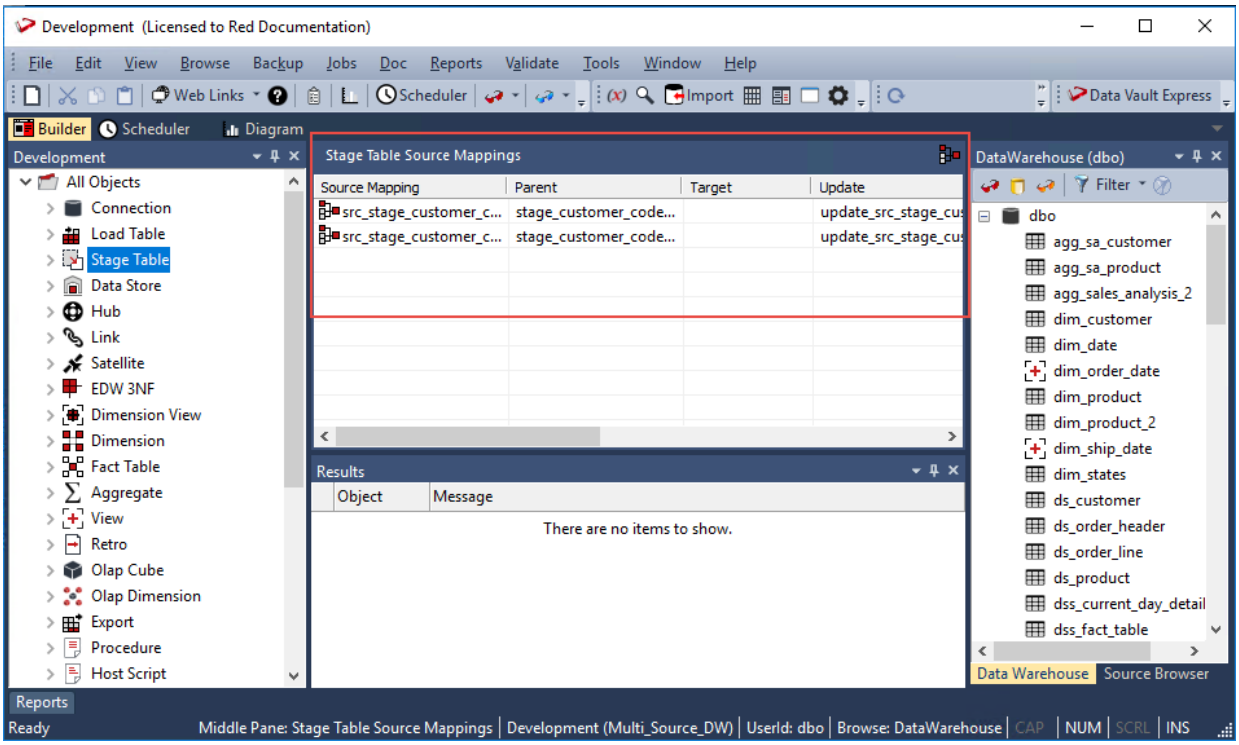
List Source Mapping Objects in an Object Group:

1. Right-click the **Object Group** from the left pane and select **List Source Mappings** from the context menu:



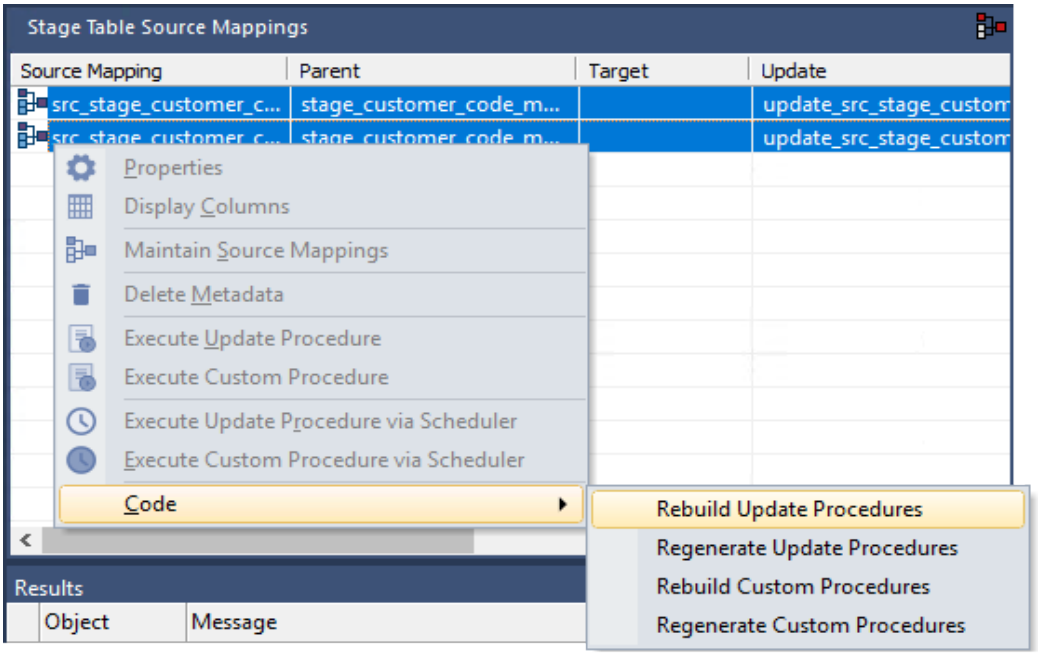
© Copyright, WhereScape Software Ltd All rights reserved.

2. The middle pane displays a list of all the source mapping objects in the selected Object Group:



Note:

Similar to other RED objects listed in the middle pane, you can select individual or multiple source mapping objects to perform operations available from the right-click context menu (e.g. rebuild update procedures for single or multiple source mapping objects).



© Copyright, WhereScape Software Ltd All rights reserved.

Table and Column Comments

WhereScape RED enables you to manage table and column comments outside your data warehouse environment and then load the updated comments back into the metadata repository; and subsequently copy them to the target database.

The comments can be exported from the metadata repository and imported back, via the **Tools > Table and Column Comments** menu option.

The **export** function enables you to output existing table or column comments into a Microsoft Excel format (.csv) file. You can open the file to view, modify or delete existing comments, as well as create new comments.

The **import** function enables you to load back the updated file into RED and perform a bulk update of the associated metadata.

The **Update Comments** context menu command enables you to copy comments from the metadata repository to the target database, e.g. updates the description and business display name fields (EUL table objects only) in the table and column **Properties** screen.

The following Object types are supported:

- Normal
- Hub
- Satellite
- Link
- Custom1
- Custom2
- Datastore
- Load
- Dimension
- Dimension View
- Stage
- Fact
- Fact KPI
- Aggregate
- Export
- Retro
- Retro Copy
- View
- Join

Notes:

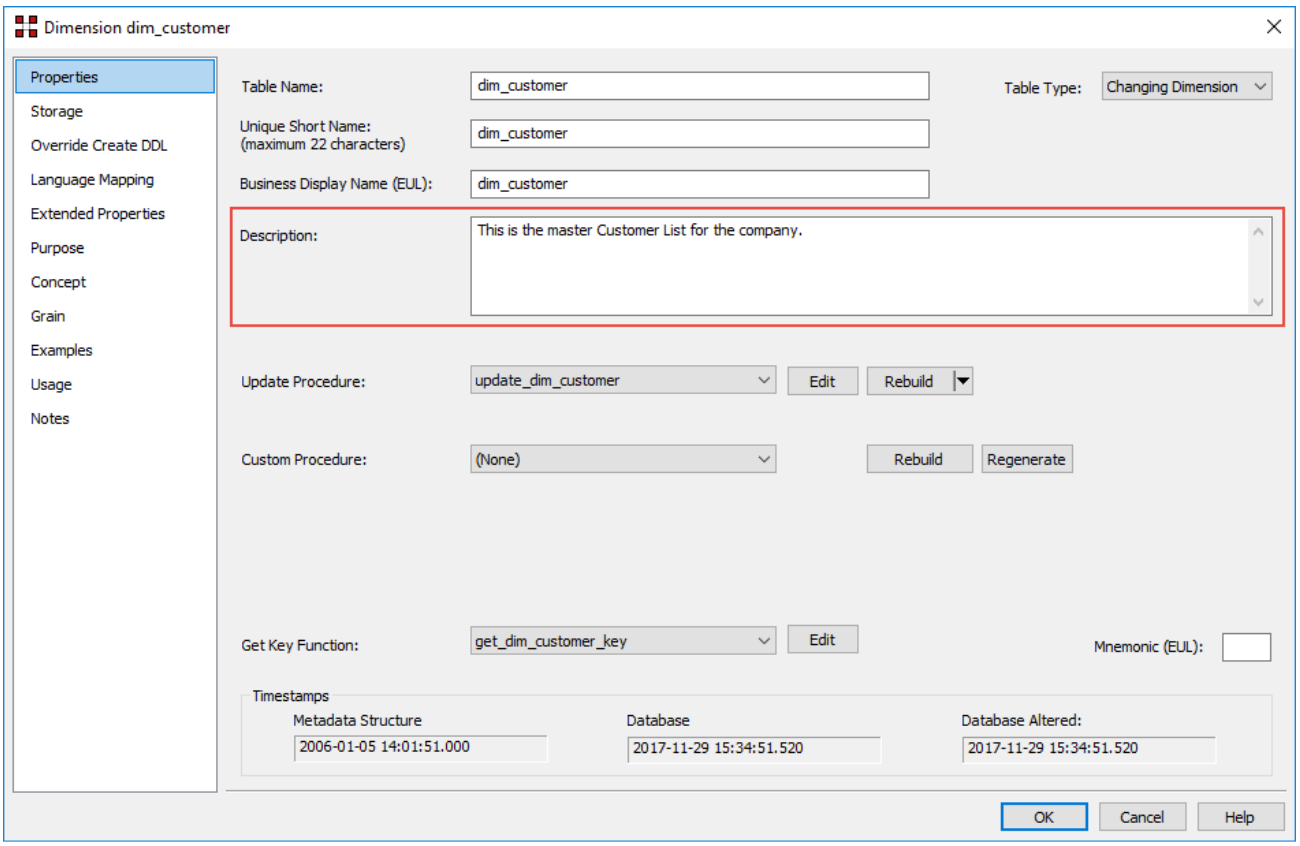
- The table and column comments export/import feature only supports UTF-8 character encoded files.
- The exported data includes the Business Display Name (EUL table objects only) which can also be updated if required.

© Copyright, WhereScape Software Ltd All rights reserved.

Defining Table Comments

Table Properties

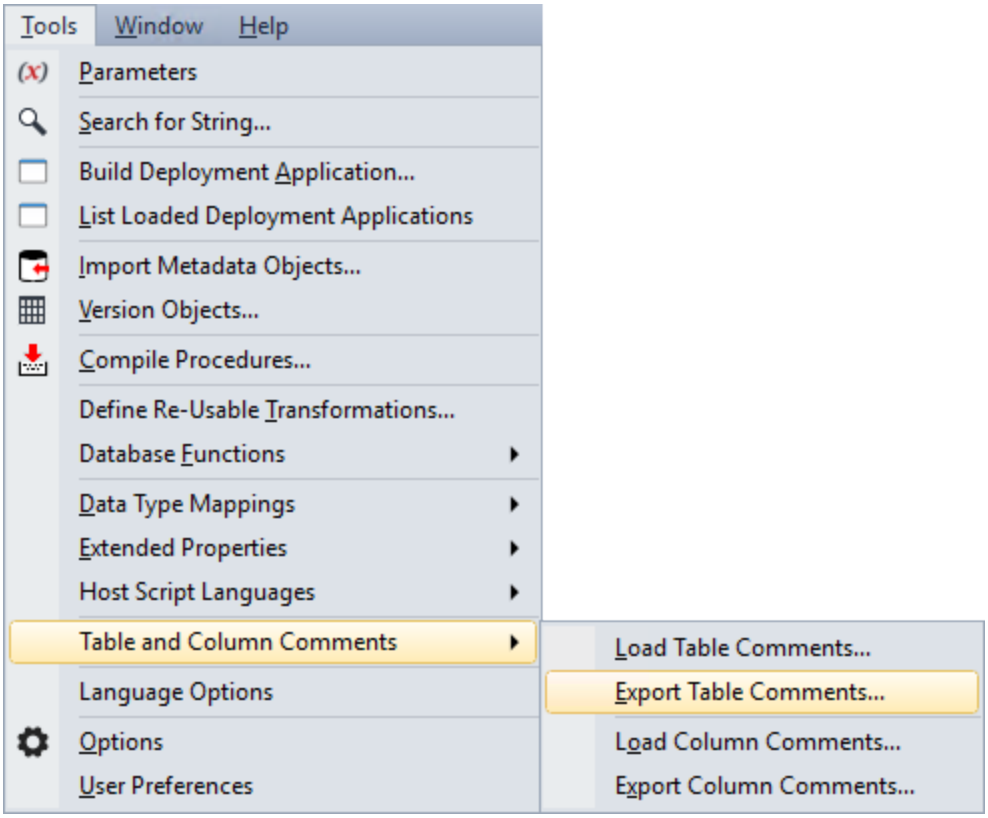
The table comments are defined in the **Description** field of the table's Properties. The comments in this field, along with the **Table Name** and **Business Display Name (EUL)** fields are used when exporting/importing table comments.



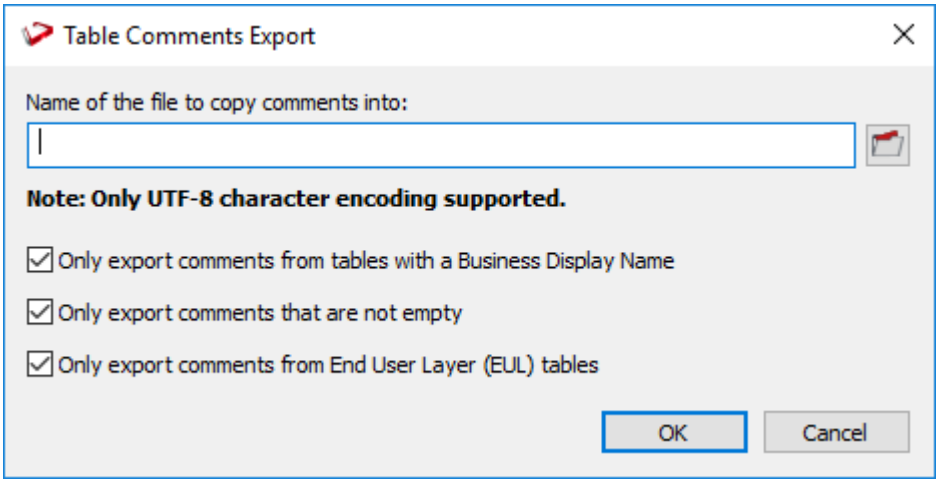
© Copyright, WhereScape Software Ltd All rights reserved.

Exporting Table Comments

To export table comments from RED, select **Tools > Table and Column Comments > Export Table Comments**.



The **Table Comments Export** pop-up window is displayed:



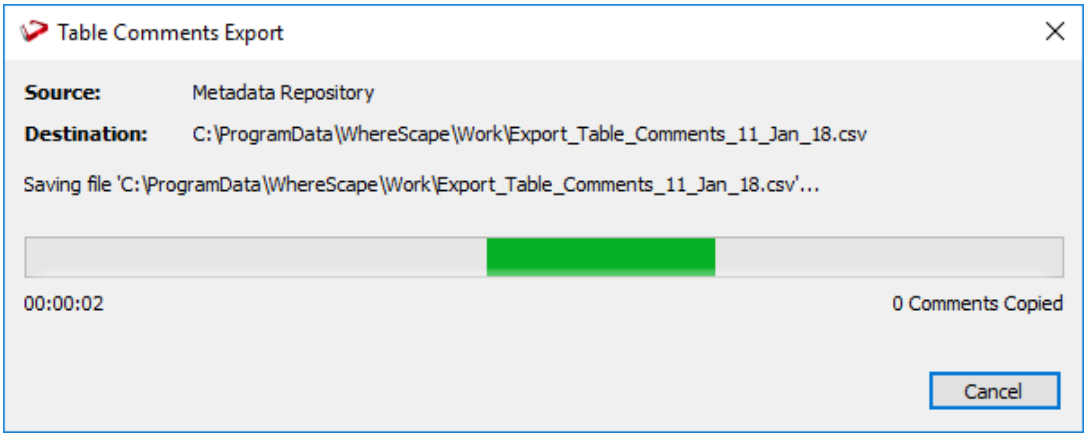
1. Type in the directory and file name of the export file or click the folder icon to navigate to the required directory.
2. Select or de-select the required export options.
    - **Only export comments from tables with a Business Display Name** - This option enables you to only process tables that have a **Business Display Name (EUL)** defined in the table's **Properties** screen. Clear the check box if you want to process all tables in your data warehouse.
    - **Only export comments that are not empty** - This option enables you to only process tables that have comments defined in the **Description** field of the table's **Properties** screen. Clear the check box if you want to process all tables in your data warehouse.

© Copyright, WhereScape Software Ltd All rights reserved.

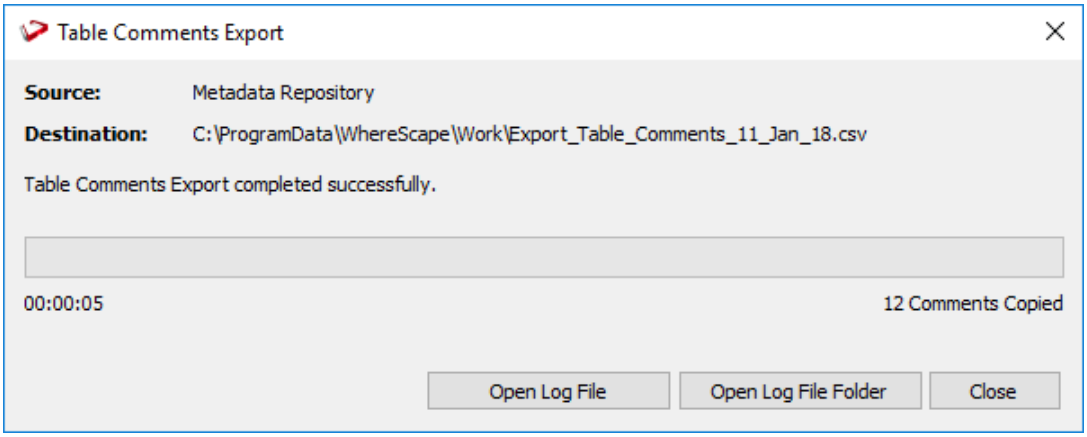
- **Only export comments from End User Layer (EUL) tables** - This option enables you to only process tables that are visible to the end user. Clear the check box if you want to process all tables in your data warehouse. The visibility of table objects to end users in RED are set in **Tools > Options > Object Types > Object Type End User Setting**, refer to **Object End User Setting** for details.

All the above options are selected by default.

3. Click **OK** to proceed with the export. The **Table Comments Export** pop-up window displays the progress of your export. You can click **Cancel** to abort the process.



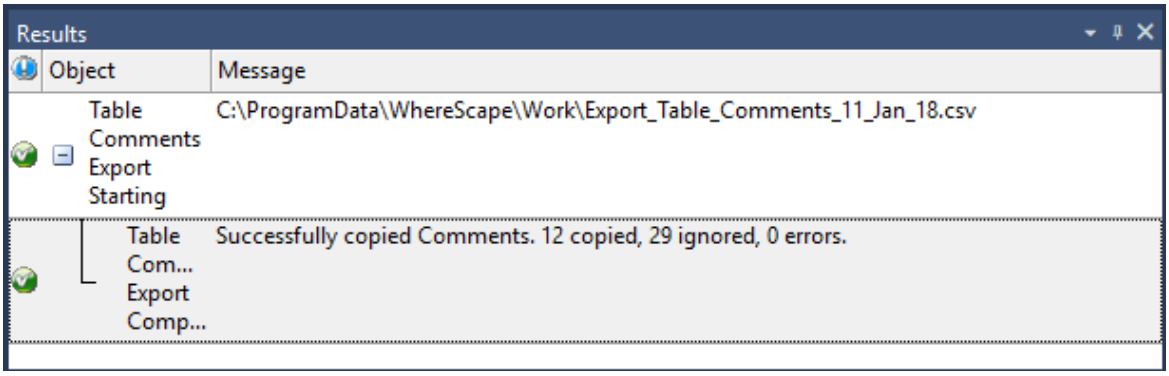
4. Once the export process is completed, the pop-window displays the processing time and the number of comments copied to the export file.



Note:

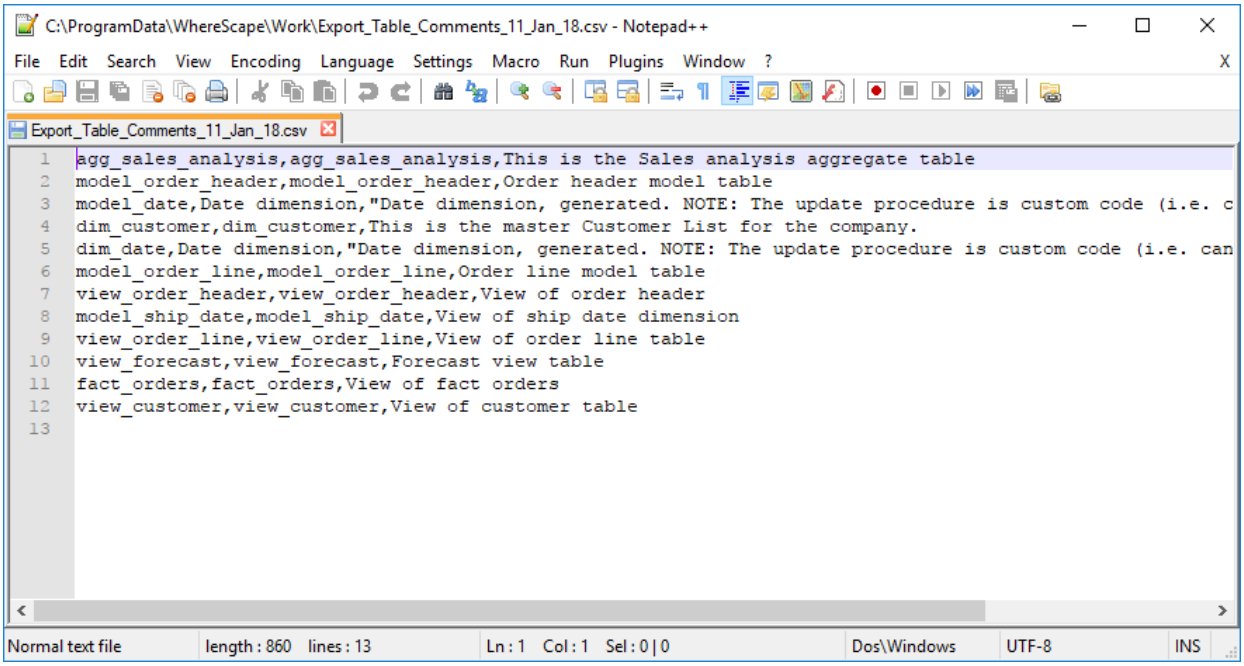
After completing the export process, you can click **Open Log File** or **Open Log File Folder** to view the log file generated. Refer to **Viewing the Import/Export Logs** for more details.

5. Click **Close** to exit the pop-up window. The **Results** pane displays a summary about the comments exported and confirms successful completion of the export.



© Copyright, WhereScape Software Ltd All rights reserved.

6. Use Microsoft Excel 2013 or Notepad to view/edit the exported comments.



Each row of data includes the Table Name, Business Display Name (EUL table objects only) and Table Comments (Description).

Notes:

The table and column comments import/export only supports UTF-8 character encoded files.

You also need to be aware of the following points, if you are going to use another tool other than Microsoft Excel 2013 to edit the comments:

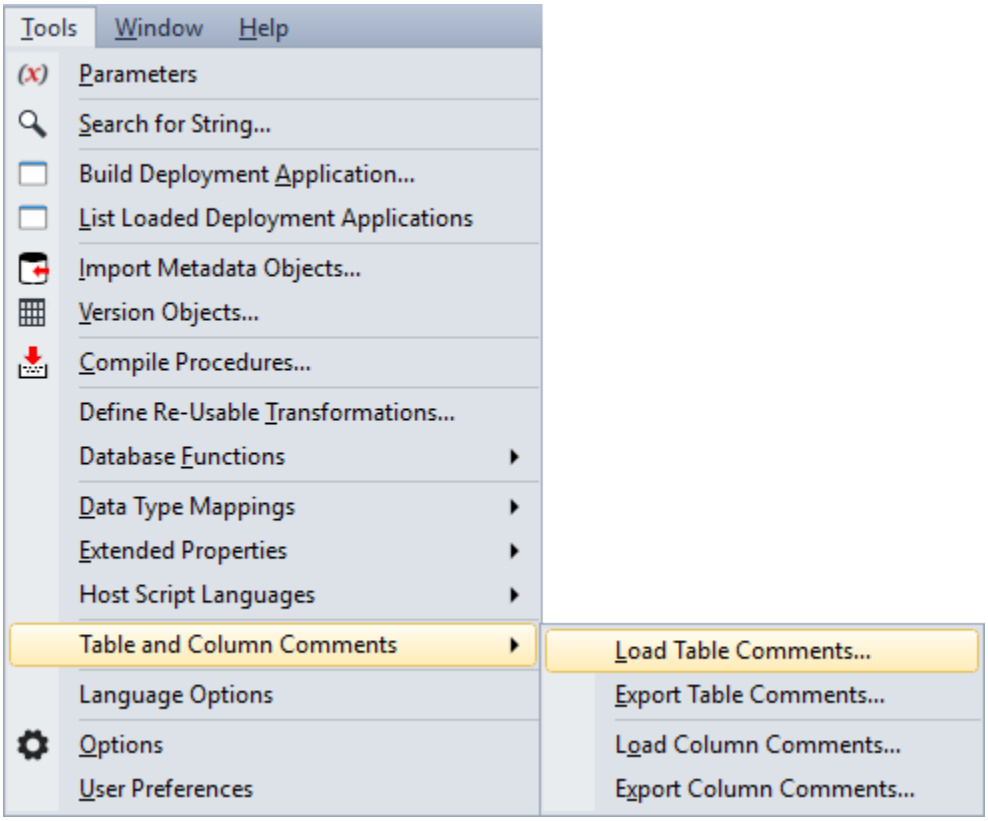
- The comment portion can be surrounded in double quotes.
- Double quotes can be escaped within the comment by prefixing the quote with another quote.
- Comma characters can be used within the comment, but only if the comment is inside double quotes.
- Comments can span multiple lines, but only if the comment is inside double quotes.

Refer to **Importing Comments from an External Source** for details.

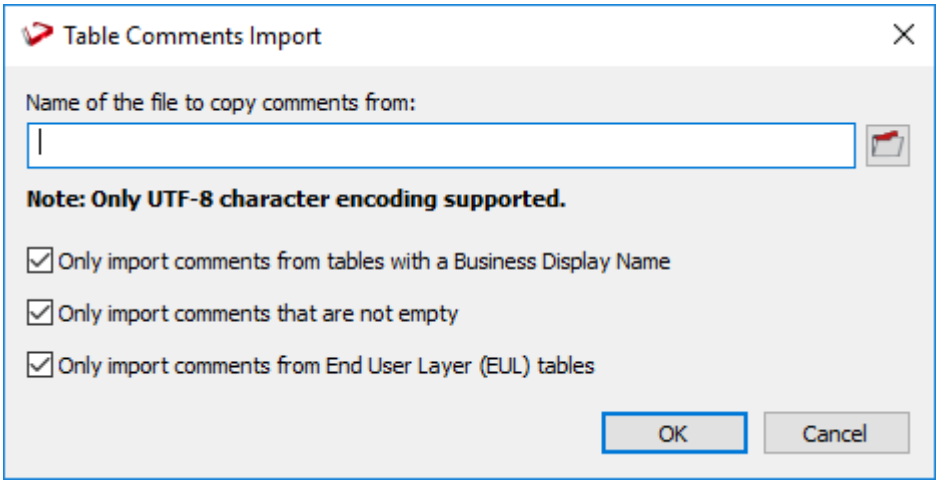
© Copyright, WhereScape Software Ltd All rights reserved.

Loading Table Comments

To load/import table comments to RED from an external (.csv) file, select **Tools > Table and Column Comments > Load Table Comments**.



The **Table Comments Import** pop-up window is displayed:



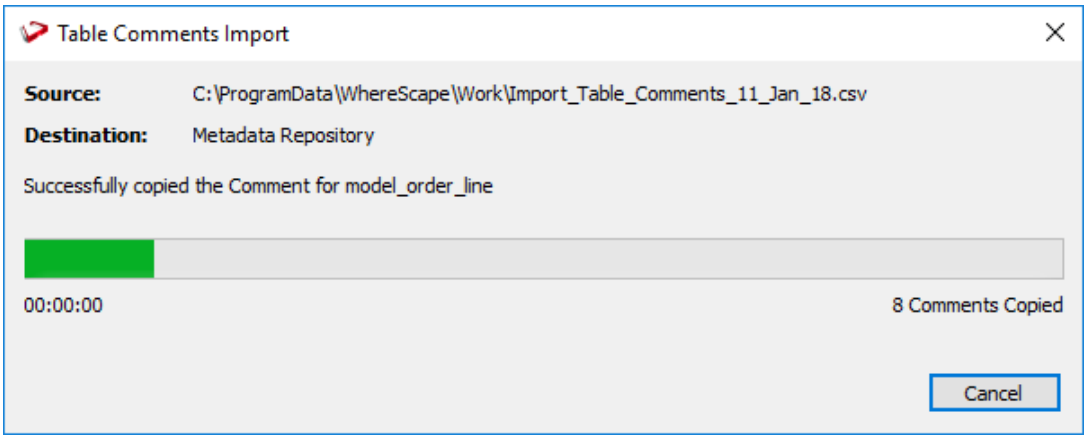
1. Type in the directory and file name of the source file or click the folder icon to navigate to the required directory.
2. Select or de-select the check box of the required import options.
    - **Only import comments from tables with a Business Display Name** - This option enables you to only process tables in the import file that have a **Business Display Name** defined. Clear the check box if you want to process all tables from your import file.
    - **Only import comments that are not empty** - This option enables you to only process tables that have comments defined in the import file. Clear the check box if you want to process all tables from your import file.
    - **Only import comments from End User Layer (EUL) tables** - This option enables you to only process tables in the import file that are visible to the end user. Clear the check box if you want

© Copyright, WhereScape Software Ltd All rights reserved.

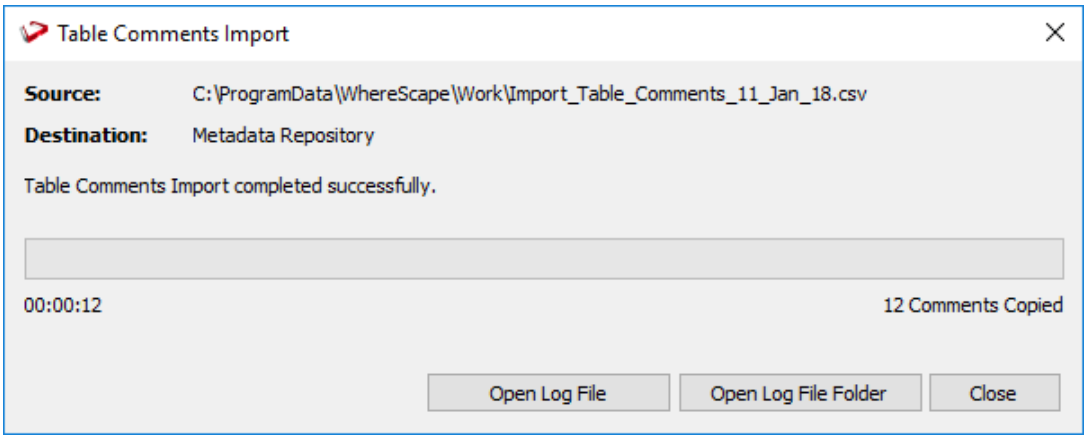
to process all tables from your import file. The visibility of table objects to end users in RED are set in **Tools > Options > Object Types > Object Type End User Setting**, refer to **Object End User Setting** for details.

All the above options are selected by default.

3. Click **OK** to proceed with the import. The **Table Comments Import** pop-up window displays the progress of your import. You can click **Cancel** to abort the process.



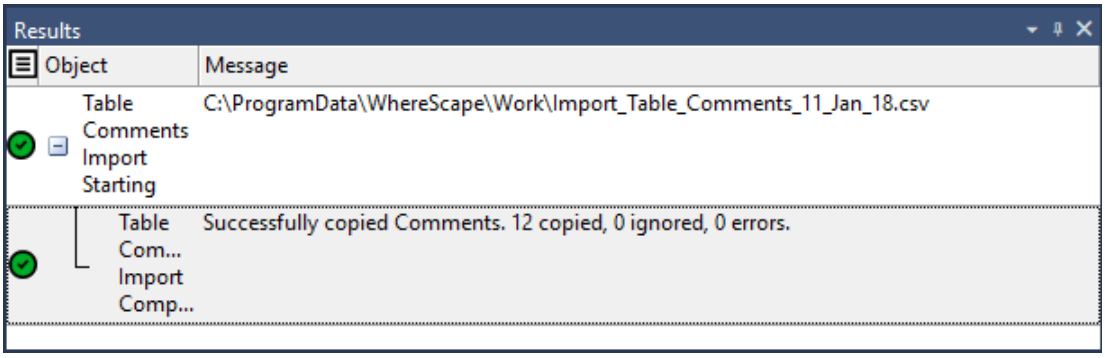
4. Once the import process is completed, the pop-window displays the processing time and the number of comments copied to the metadata repository.



Note:

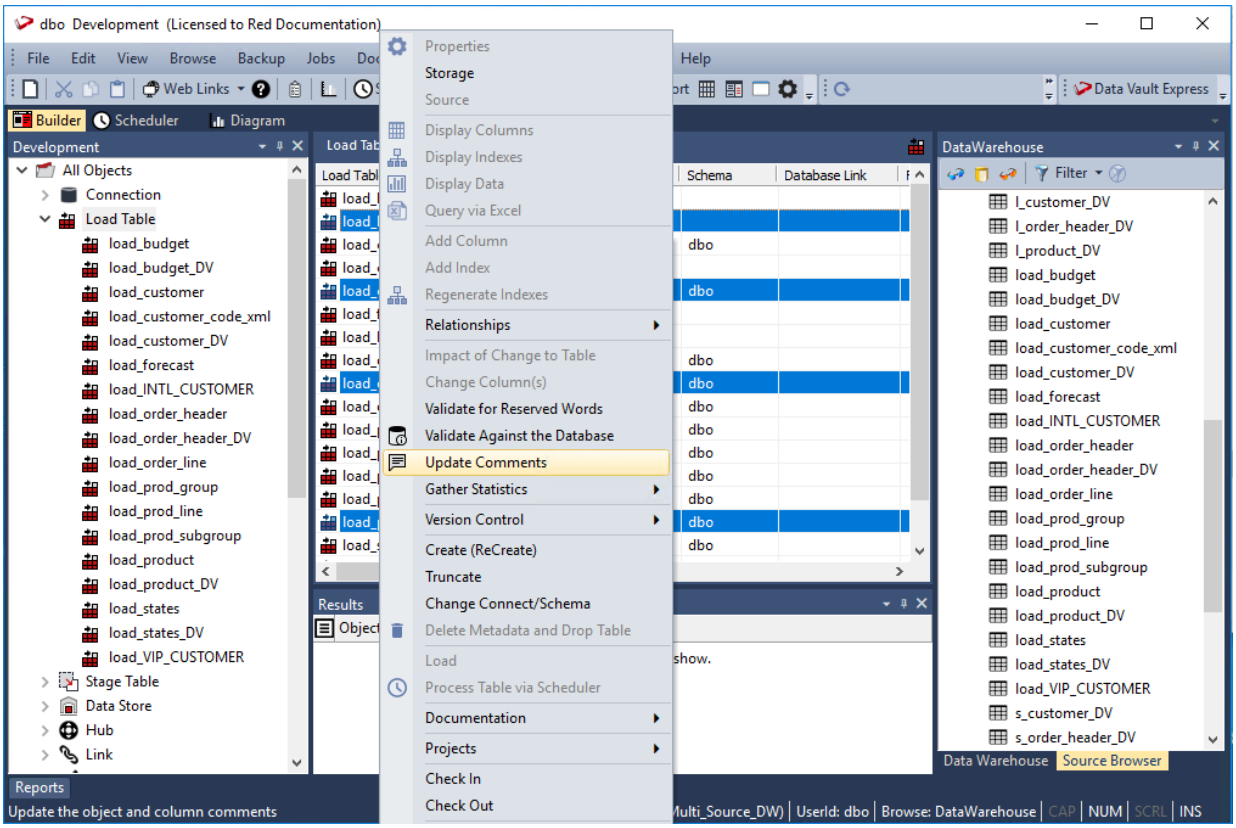
After completing the import process, you can click **Open Log File** or **Open Log File Folder** to view the log file generated. Refer to **Viewing the Import/Export** Logs for details.

5. Click **Close** to exit the pop-up window. The **Results** pane displays a summary about the comments imported and confirms successful completion of the import.

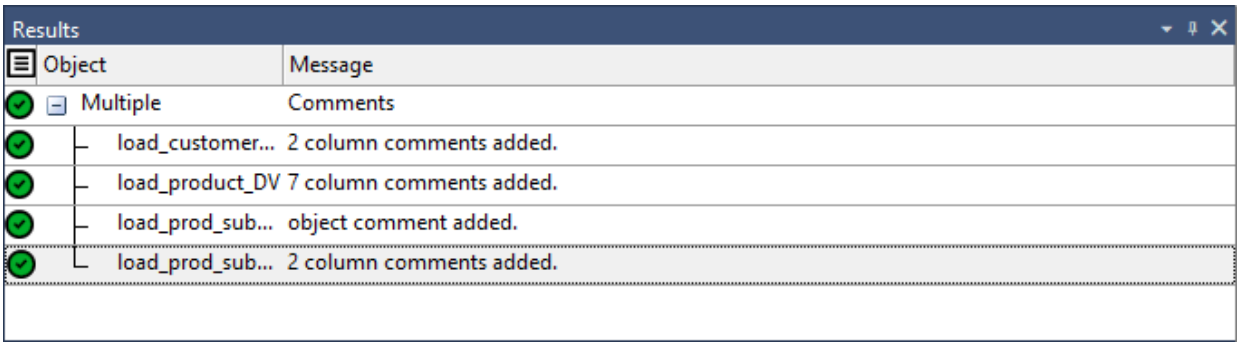


6. Use the **Update Comments** command from the context menu to copy the comments from the metadata repository to the corresponding tables in the target database.

© Copyright, WhereScape Software Ltd All rights reserved.



The **Results** pane displays the table(s) and column(s) that have added comments.



Notes:

The table and column comments import/export only supports UTF-8 character encoded files. You also need to be aware of the following points, if you are going to use another tool other than Microsoft Excel 2013 to edit the comments:

- The comment portion can be surrounded in double quotes.
- Double quotes can be escaped within the comment by prefixing the quote with another quote.
- Comma characters can be used within the comment, but only if the comment is inside double quotes.
- Comments can span multiple lines, but only if the comment is inside double quotes.

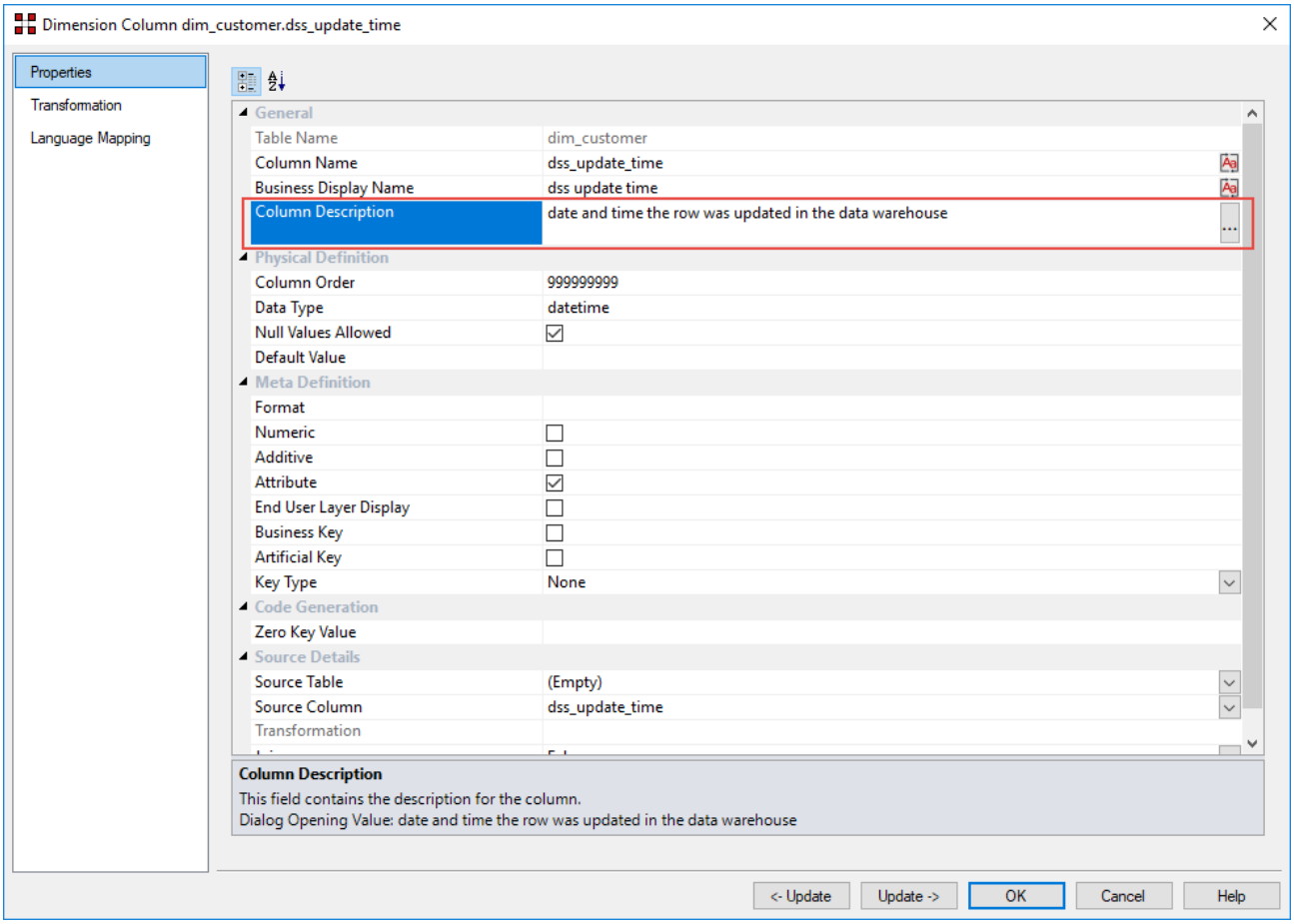
Refer to **Importing Comments from an External Source** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

Defining Column Comments

Column Properties

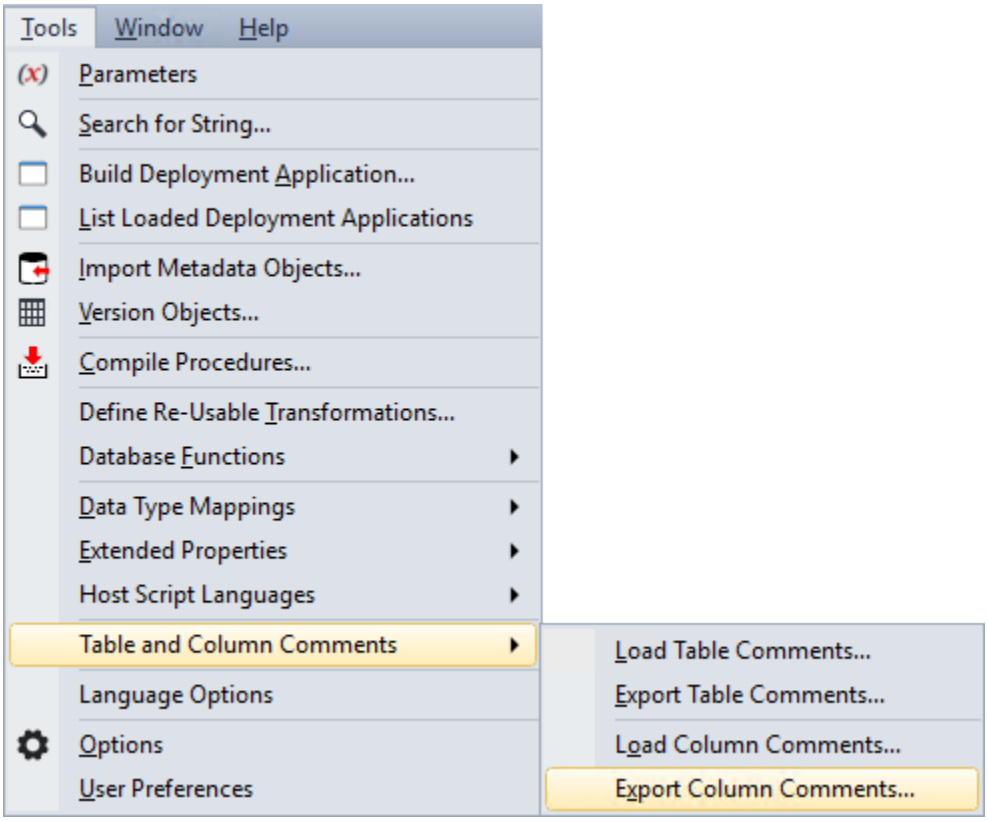
The column comments are defined in the **Column Description** field of the column's **Properties**. The comments in this field, along with the **Table Name**, **Column Name**, and **Column Title/Business Display Name** fields are used when exporting/importing column comments.



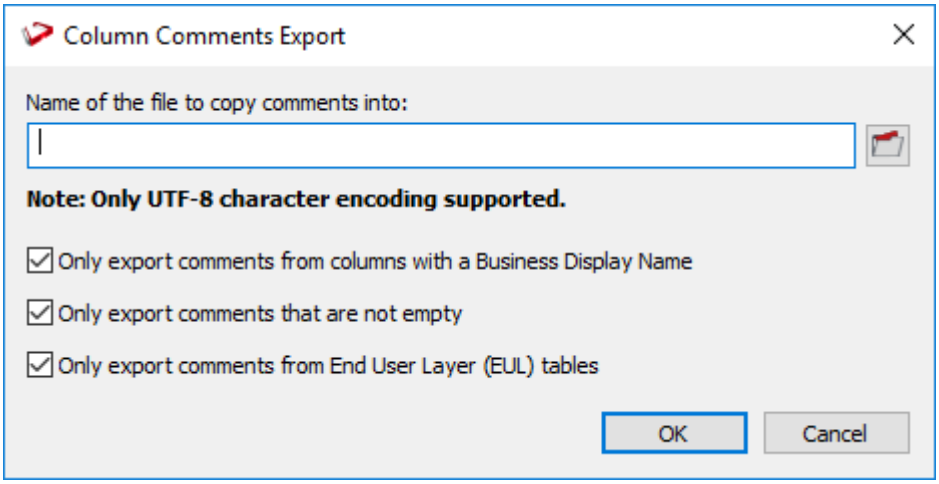
© Copyright, WhereScape Software Ltd All rights reserved.

Exporting Column Comments

To export table comments from RED, select **Tools > Table and Column Comments > Export Column Comments**.



The **Column Comments Export** pop-up window is displayed:



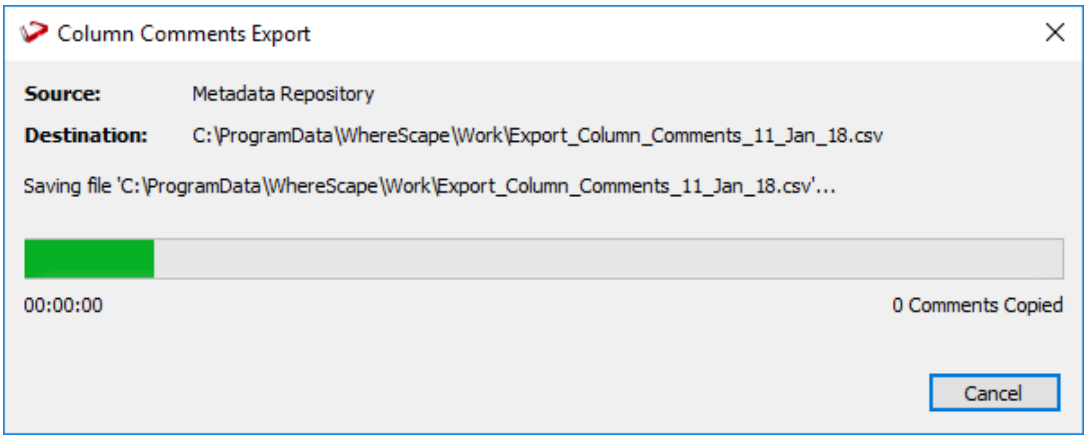
1. Type in the directory and file name of the export file or click the folder icon to navigate to the required directory.
2. Select or de-select the required export options.
    - **Only export comments from columns with a Business Display Name** - This option enables you to only process columns that have a **Business Display Name (Column Title)** defined in the column's **Properties** screen. Clear the check box if you want to process all columns in your data warehouse.
    - **Only export comments that are not empty** - This option enables you to only process columns that have comments defined in Column Description field of the column's **Properties** screen. Clear the check box if you want to process all columns in your data warehouse.
    - **Only export comments from End User Layer (EUL) tables** - This option enables you to only process columns from tables that are visible to the end user. Clear the check box if you want to

© Copyright, WhereScape Software Ltd All rights reserved.

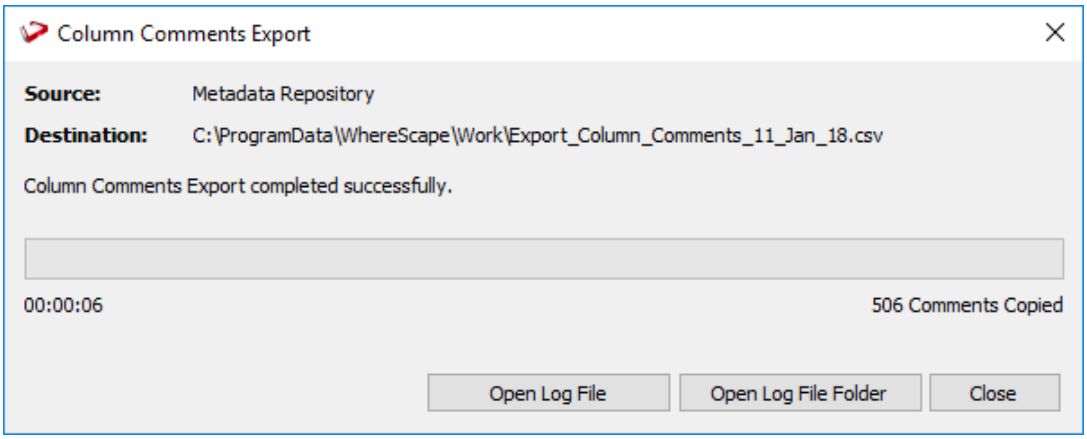
process all columns in your data warehouse.
The visibility of table objects to end users in RED are set in **Tools > Options > Object Types > Object Type End User Setting**, refer to **Object End User Setting** for details.

All the above options are selected by default.

3. Click **OK** to proceed with the export. The **Column Comments Export** pop-up window displays the progress of your export. You can click **Cancel** to abort the process.



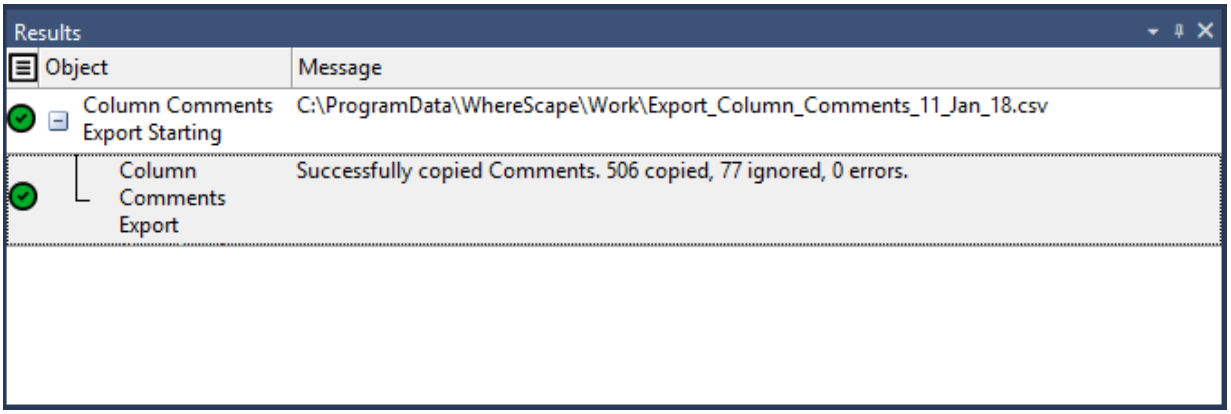
4. Once the export process is completed, the pop-window displays the processing time and the number of comments copied to the export file.



Note

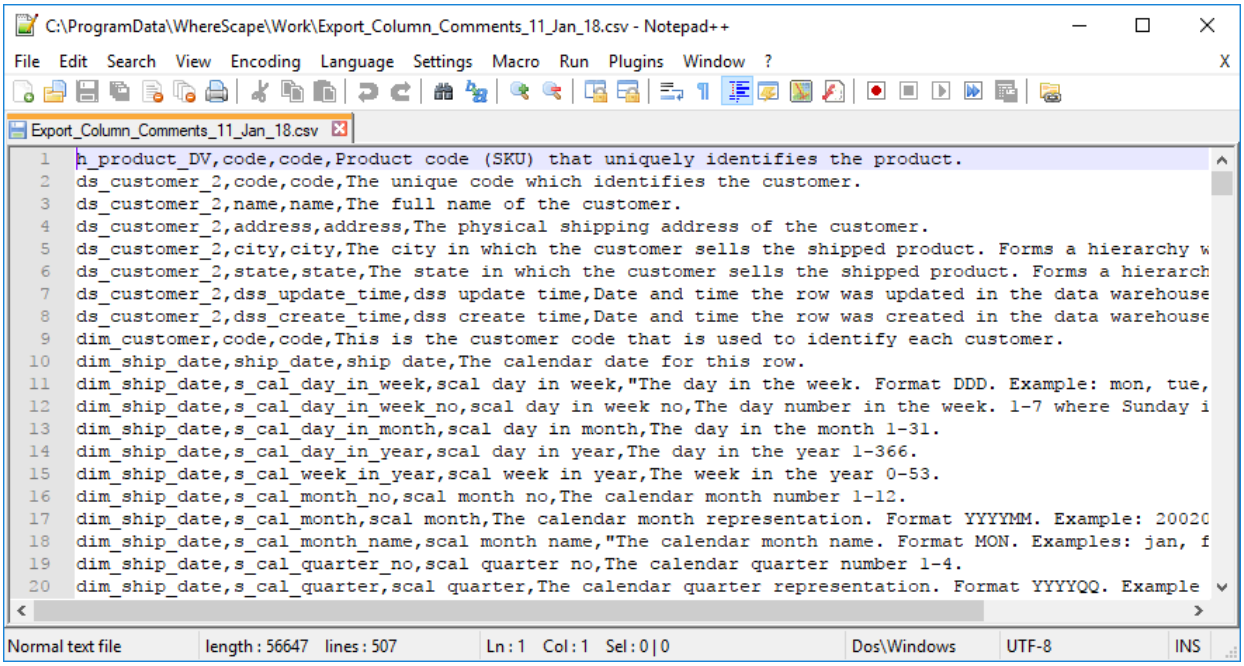
After completing the export process, you can click **Open Log File** or **Open Log File Folder** to view the log file generated. Refer to **Viewing the Import/Export Logs** for more details.

5. Click **Close** to exit the pop-up window. The **Results** pane displays a summary about the comments exported and confirms successful completion of the export.



© Copyright, WhereScape Software Ltd All rights reserved.

6. Use Microsoft Excel 2013 or Notepad to view/edit the exported comments.



Each row of data includes the Table Name, Column Name, Column Title/Business Display Name and Column Comments (Description).

Notes

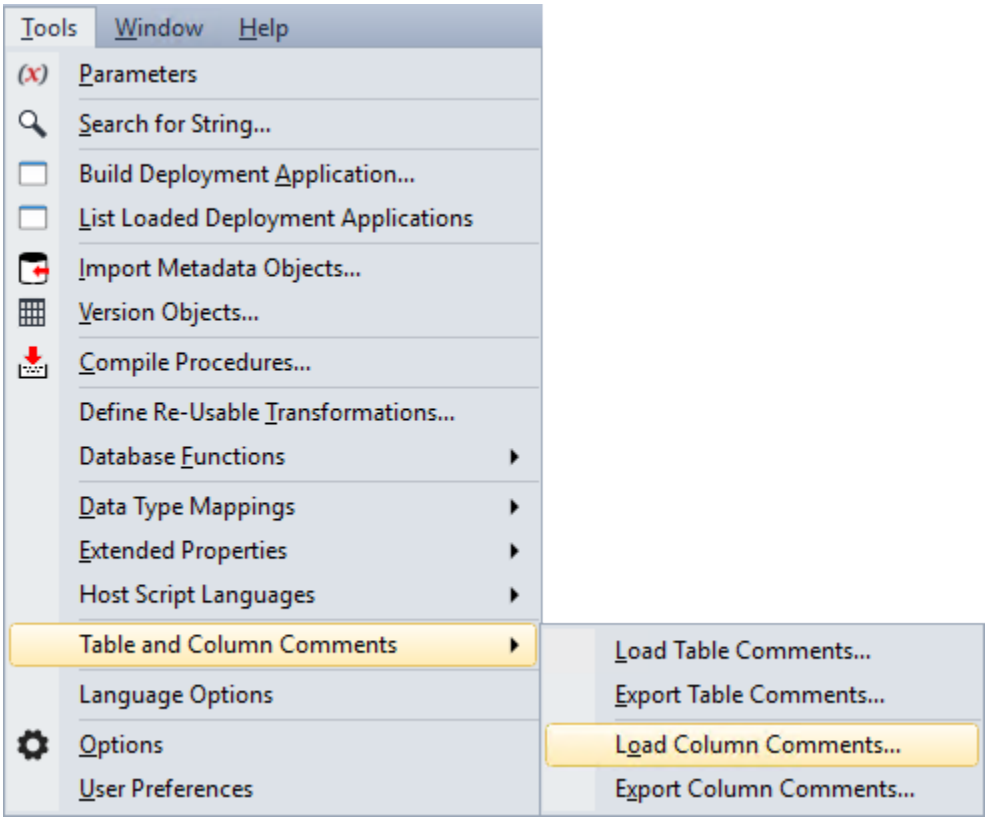
The table and column comments import/export only supports UTF-8 character encoded files.
You also need to be aware of the following points, if you are going to use another tool other than Microsoft Excel 2013 to edit the comments:

- The comment portion can be surrounded in double quotes.
- Double quotes can be escaped within the comment by prefixing the quote with another quote.
- Comma characters can be used within the comment, but only if the comment is inside double quotes.
- Comments can span multiple lines, but only if the comment is inside double quotes.
- Refer to **Importing Comments from an External Source** for details.

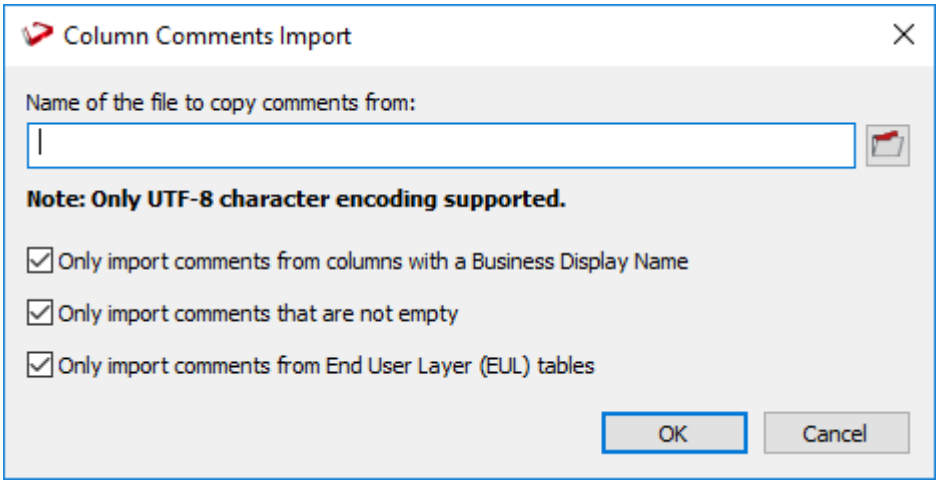
© Copyright, WhereScape Software Ltd All rights reserved.

Loading Column Comments

To load/import column comments to RED from an external (.csv) file, select **Tools > Table and Column Comments > Load Column Comments**.



The **Column Comments Import** pop-up window is displayed:



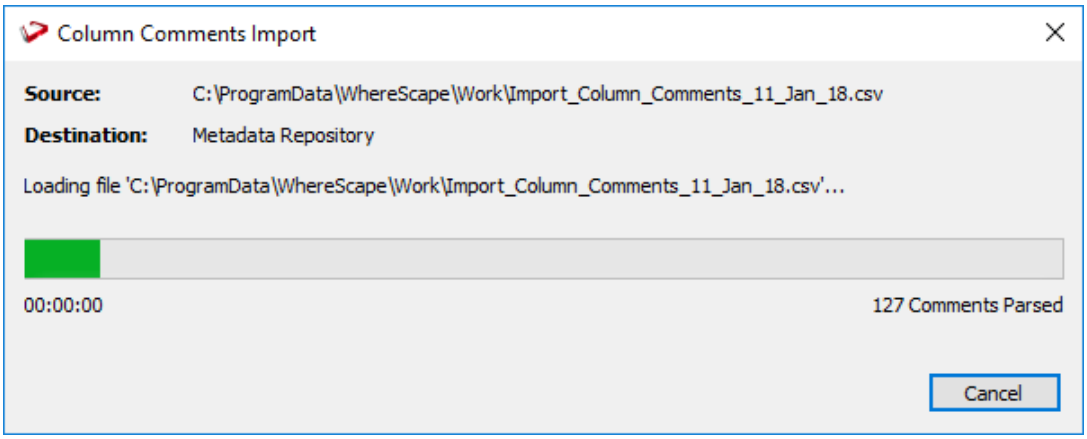
1. Type in the directory and file name of the source file or click the folder icon to navigate to the required directory.
2. Select or de-select the check box of the required import options.
    - **Only import comments from tables with a Business Display Name** - This option enables you to only process columns in the import file that have a Business Display Name/Column Title defined. Clear the check box if you want to process all tables from your import file.
    - **Only import comments that are not empty** - This option enables you to only process columns that have comments defined in the import file. Clear the check box if you want to process all tables from your import file.
    - **Only import comments from End User Layer (EUL) tables** - This option enables you to only process columns from tables in the import file that are visible to the end user. Clear the check

© Copyright, WhereScape Software Ltd All rights reserved.

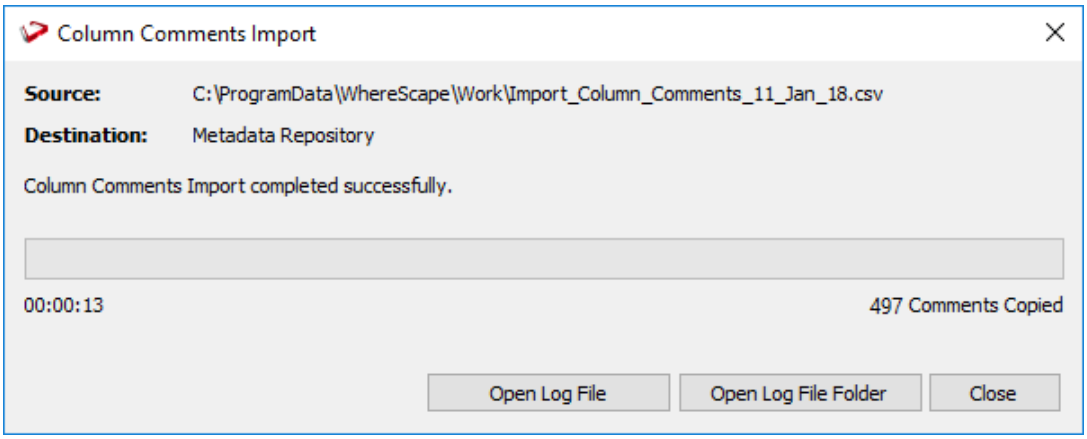
box if you want to process all tables from your import file. The visibility of table objects to end users in RED are set in **Tools > Options > Object Types > Object Type End User Setting**, refer to **Object End User Setting** for details.

All the above options are selected by default.

3. Click **OK** to proceed with the import. The **Column Comments Import** pop-up window displays the progress of your import. You can click **Cancel** to abort the process.



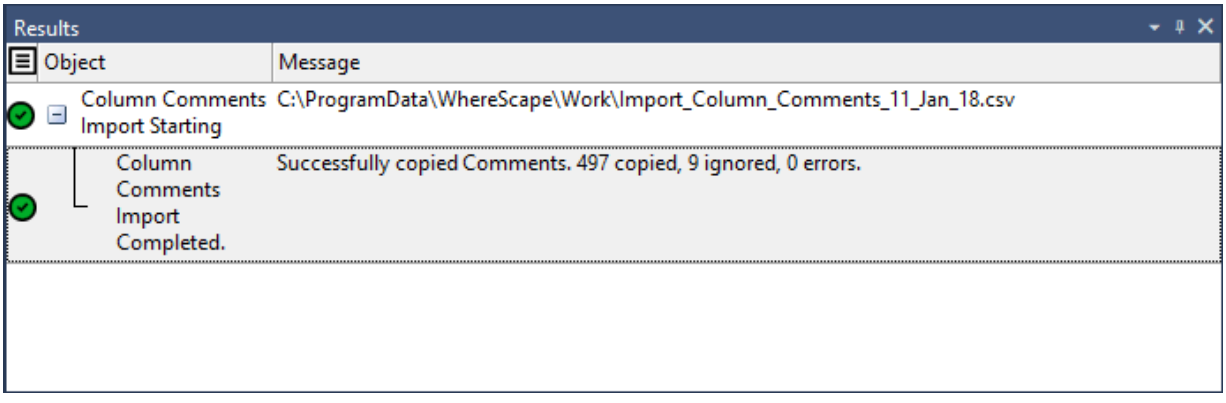
4. Once the import process is completed, the pop-window displays the processing time and the number of comments copied to the metadata repository.



Note

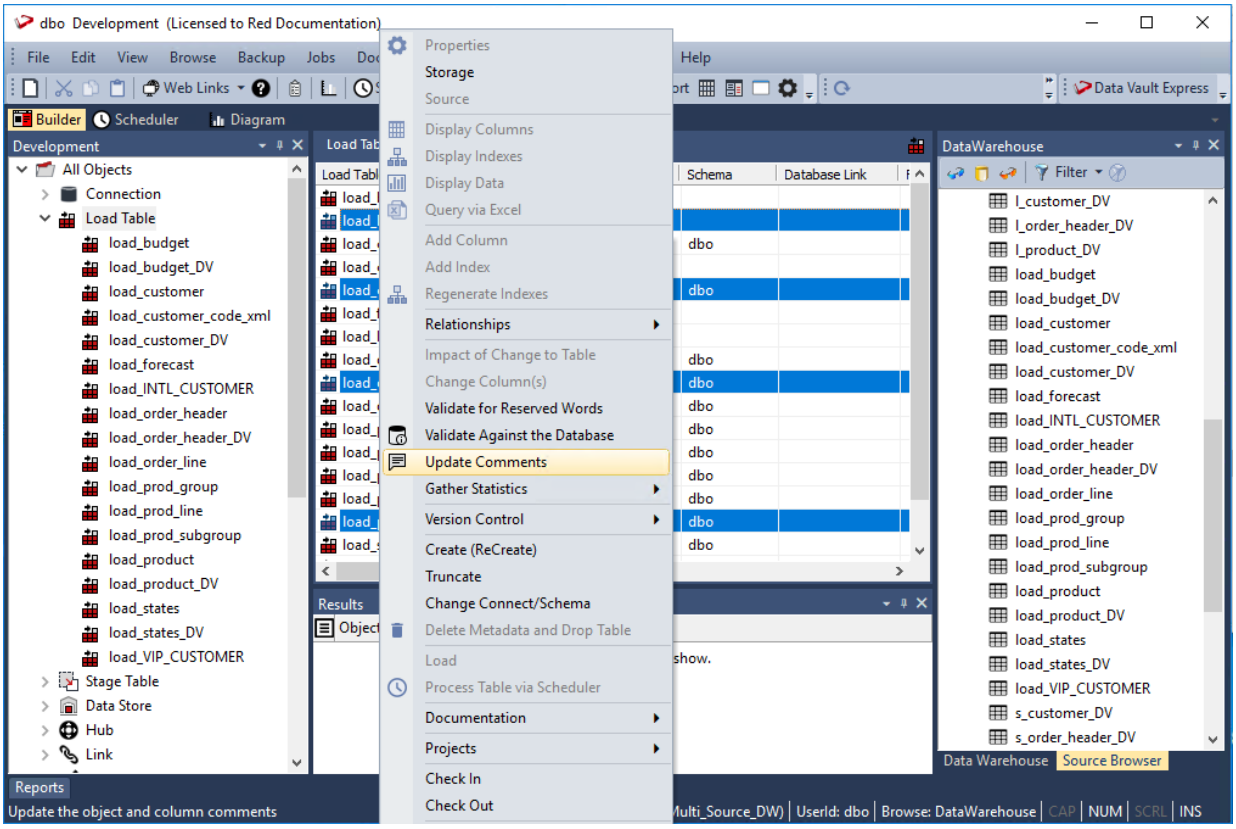
After completing the import process, you can click **Open Log File** or **Open Log File Folder** to view the log file generated. Refer to **Viewing the Import/Export Logs** for details.

5. Click **Close** to exit the pop-up window. The **Results** pane displays a summary about the comments imported and confirms successful completion of the import.

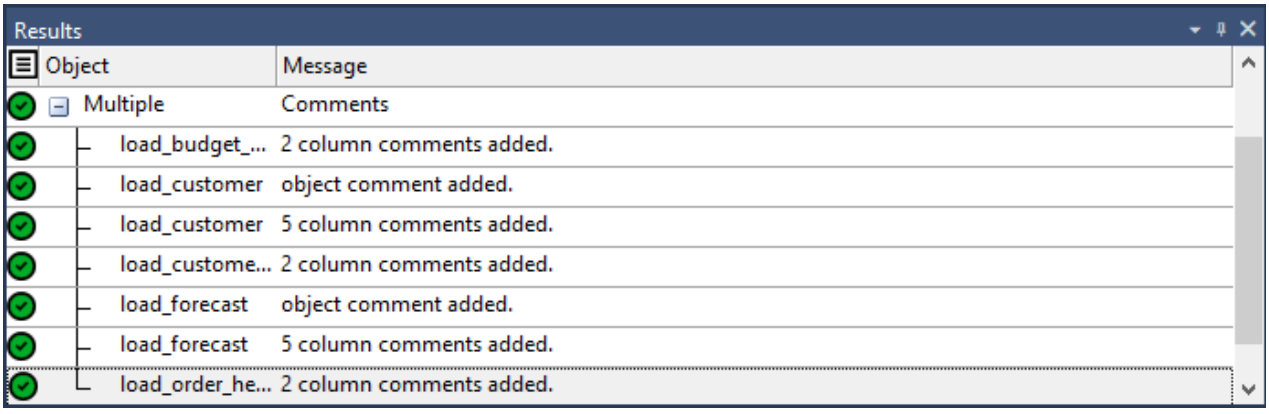


© Copyright, WhereScape Software Ltd All rights reserved.

6. Use the **Update Comments** command from the context menu, to copy the comments from the metadata repository to the corresponding columns in the target database.



The **Results** pane displays the table(s) and column(s) that have added comments.



Notes

The table and column comments import/export only supports UTF-8 character encoded files. You also need to be aware of the following points, if you are going to use another tool other than Microsoft Excel 2013 to edit the comments:

- The comment portion can be surrounded in double quotes.
- Double quotes can be escaped within the comment by prefixing the quote with another quote.
- Comma characters can be used within the comment, but only if the comment is inside double quotes.
- Comments can span multiple lines, but only if the comment is inside double quotes.
- Refer to **Importing Comments from an External Source** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

Importing Comments from an External Source

The import function supports Microsoft Excel 2013 comma separated value (.csv) file format for importing comments to RED.

Table Comments

Each row of data includes the Table Name, Business Display Name (EUL table objects only) and Table Comments (Description).

```
view_order_header,view_order_header,View of order header.
view_order_line,view_order_line,View of order line table.
view_forecast,view_forecast,View of Forecast table.
```

Column Comments

Each row of data includes the Table Name, Column Name, Column Title/Business Display Name and Column Comments (Description).

```
view_order_header,order_number,order_number,A code which identifies the order.
view_order_line,order_line_no,order_line_no,A number which identifies a line on the order.
view_forecast,forecast_quantity,forecast_quantity,Quantity of product forecasted (i.e.
number of product units forecasted to be sold).
```

Notes

- The comment portion can be surrounded in double quotes. For example:

```
view_order_header,view_order_header,"View of order header."
view_order_line,view_order_line,"View of order line table."
view_forecast,view_forecast,"View of Forecast table."
```

- Double quotes can be escaped within the comment by prefixing the quote with another quote. For example:

```
view_order_header,view_order_header,"View ""of"" order header."
view_order_line,view_order_line,View of order line table.
view_forecast,view_forecast,View of Forecast table.
```

- Comma characters can be used within the comment, but only if the comment is inside double quotes. For example:

```
view_order_header,view_order_header,"View of order header,comment."
view_order_line,view_order_line,"View of order line table,comment."
view_forecast,view_forecast,"View of Forecast table,comment."
```

- Comments can span multiple lines, but only if the comment is inside double quotes. For example:

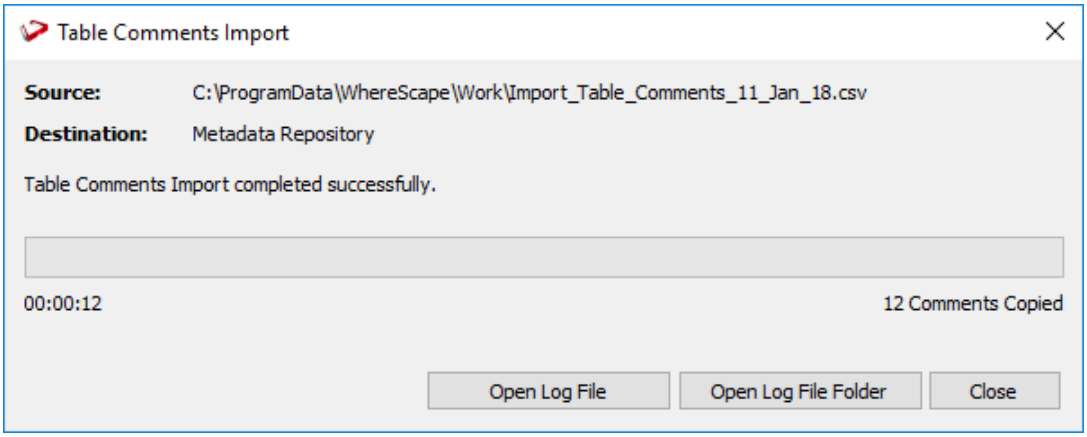
```
view_order_header,view_order_header,"View of order header comment."
view_order_line,view_order_line,"View of order line
table comment."
view_forecast,view_forecast,"View of Forecast
table comment."
```

© Copyright, WhereScape Software Ltd All rights reserved.

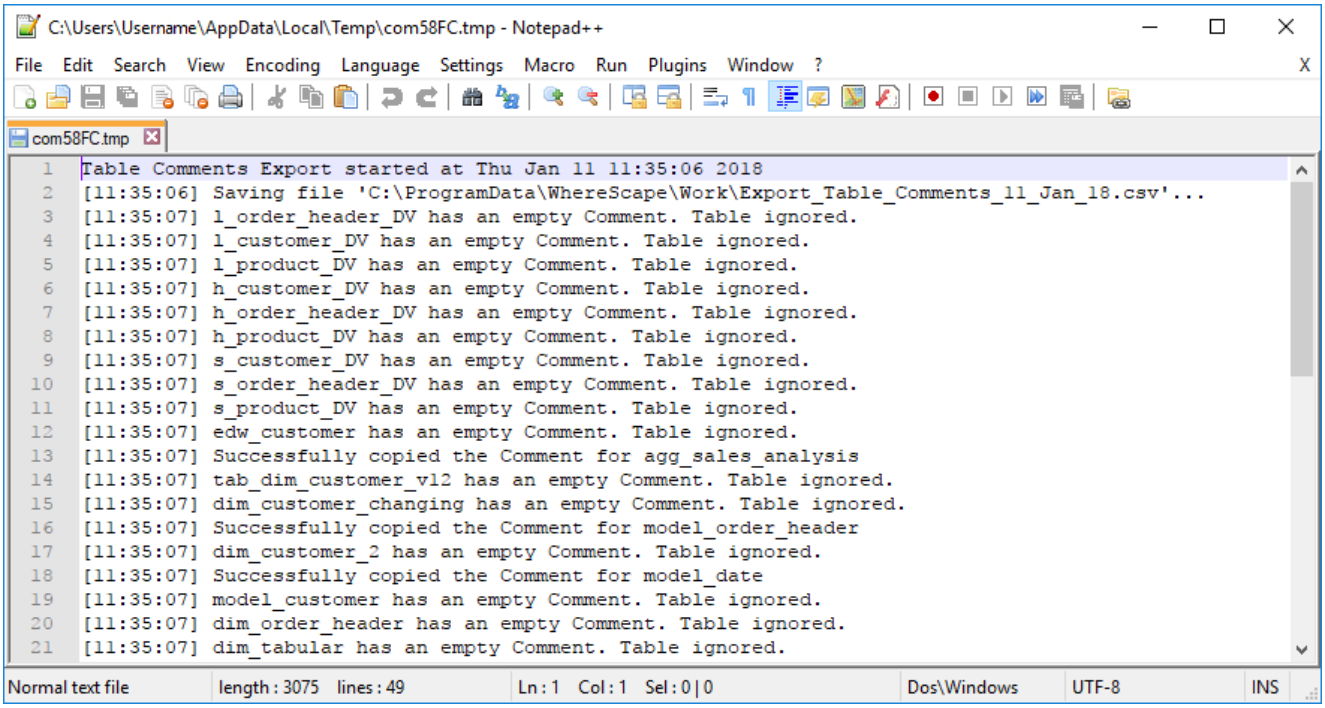
Viewing the Import/Export Logs

RED creates a log to aid in troubleshooting, in case issues are encountered during the import/export process.

After completing the import/export process, you can click **Open Log File** or **Open Log File Folder** from the **Table Comments Import/Export** pop-up window to view the log file generated.



The import/export logs are stored in your Windows Temp directory (`c:\Users\Username\AppData\Local\Temp`).



© Copyright, WhereScape Software Ltd All rights reserved.

Host Script Languages

WhereScape RED provides a facility for defining and managing host script languages to enable users to implement host scripts in their preferred scripting language, e.g. Python, Perl, PowerShell, etc.

This provide RED users with the flexibility of using different types of script languages for the various script-based processing that they perform within RED, such as the script based update of the data warehouse content, created in custom database targets. Refer to **Script Templates for Custom Database Table Objects** for details.

This feature also supports importing a set of host script language definitions into RED from an external source, and exporting host script languages defined in RED to a file. This enables you to migrate and reuse host script language definitions between different repositories.

Using the host script languages feature involves defining the language properties which includes the command to run and execute a script in the defined language.

The host script language definitions are created and maintained in the **Maintain Host Script Languages** screen.

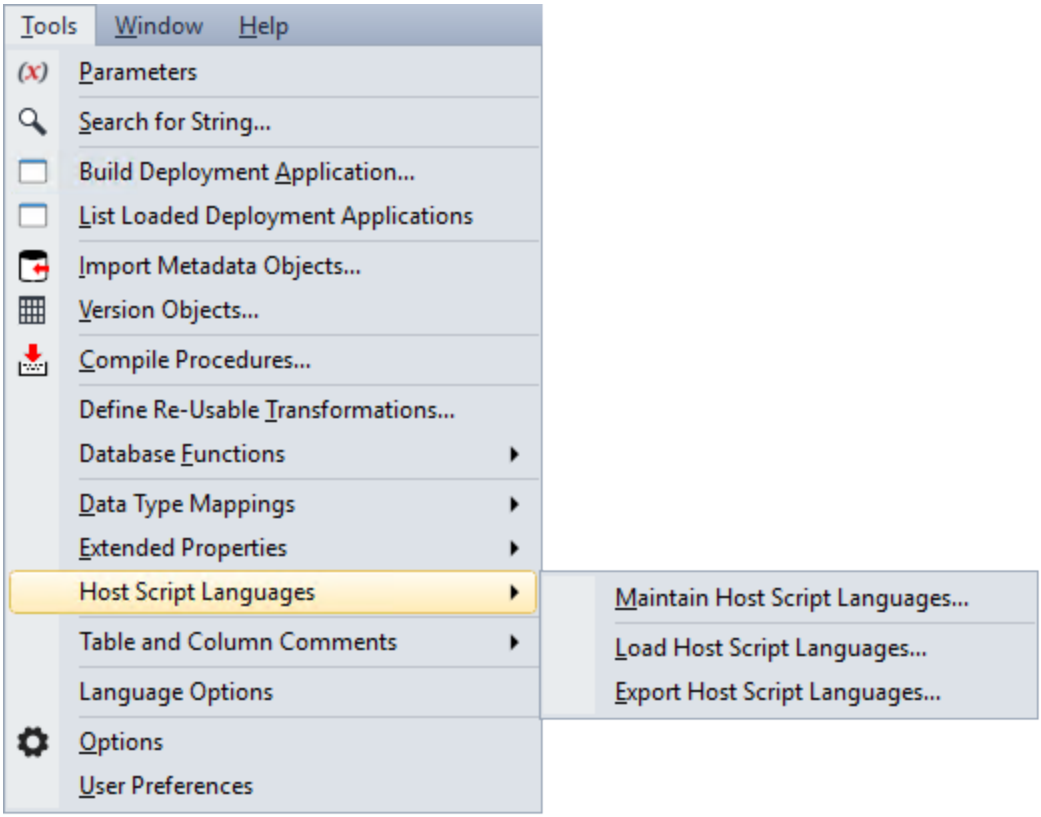
© Copyright, WhereScape Software Ltd All rights reserved.

Defining Host Script Languages

Host Script Language definitions consist of the following :

Property	Description
Name	The name for the host script language, as displayed in the drop-down fields for selecting the script type (e.g. **Type** field in the **Host Script** and **Template** Properties screens—see **Applying Host Script Languages**). This name must be consistent and match across repositories, to preserve the associations with the host script language when deploying applications.
Description	The description of the host script language which is for display purposes only.
File Extension	The file extension used when the scripts are written to files, for example **py** for Python script files. If this is left blank, the generated files will have no extension.
Command	The command to execute a script in the defined language. The token $SCRIPT_NAME$ is replaced with the full path and file name of the script file to execute. The substituted command is used as the arguments for the PowerShell call operator (&). For example, "C:\Program Files\Python37\python.exe" "$SCRIPT_NAME". The defined path must be valid for all RED clients and schedulers that are required to execute scripts for the defined host script language.

These user defined host script languages can be created and managed, using the **Host Script Languages** facility which is launched from the **Tools** menu.



The following options are available:

- **Maintain Host Script Languages** is used to add new or edit existing host script language definition settings.
- **Load Host Script Languages** to load a set of host script language definitions into RED from an external source. This option is typically used with a WhereScape provided enablement pack.
- **Export Host Script Languages** is used to export the available host script languages in RED to a host script language file (.hsclang).

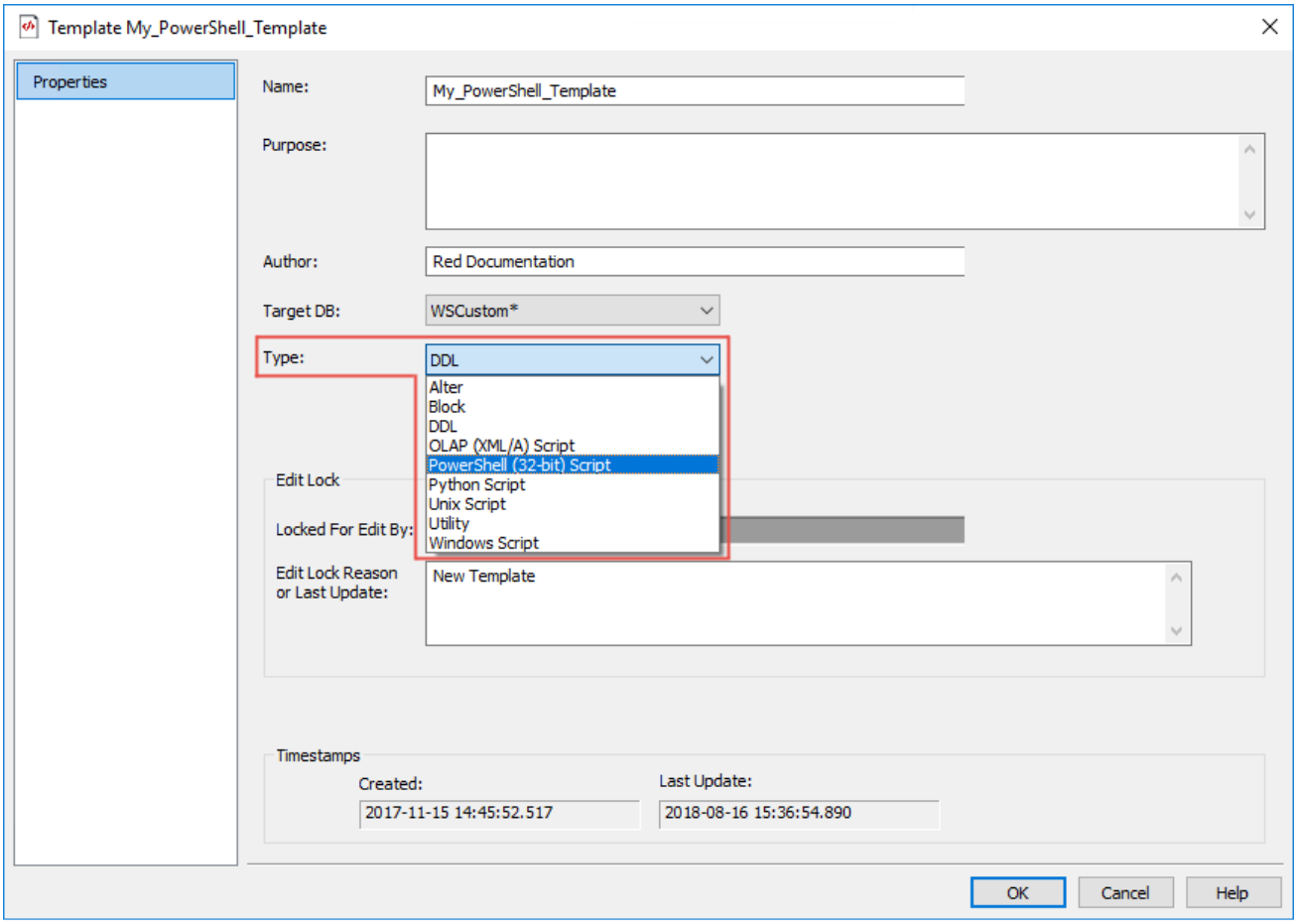
© Copyright, WhereScape Software Ltd All rights reserved.

Applying Host Script Languages

The defined Host Script Language types can be applied as a Template type or as the type of a Host Script object in RED. This is done via the **Properties** screen of the Template or Host Script object.

Template Properties

The **Type** field specifies what the template can be used for in RED. Refer to **Template Properties** for details.

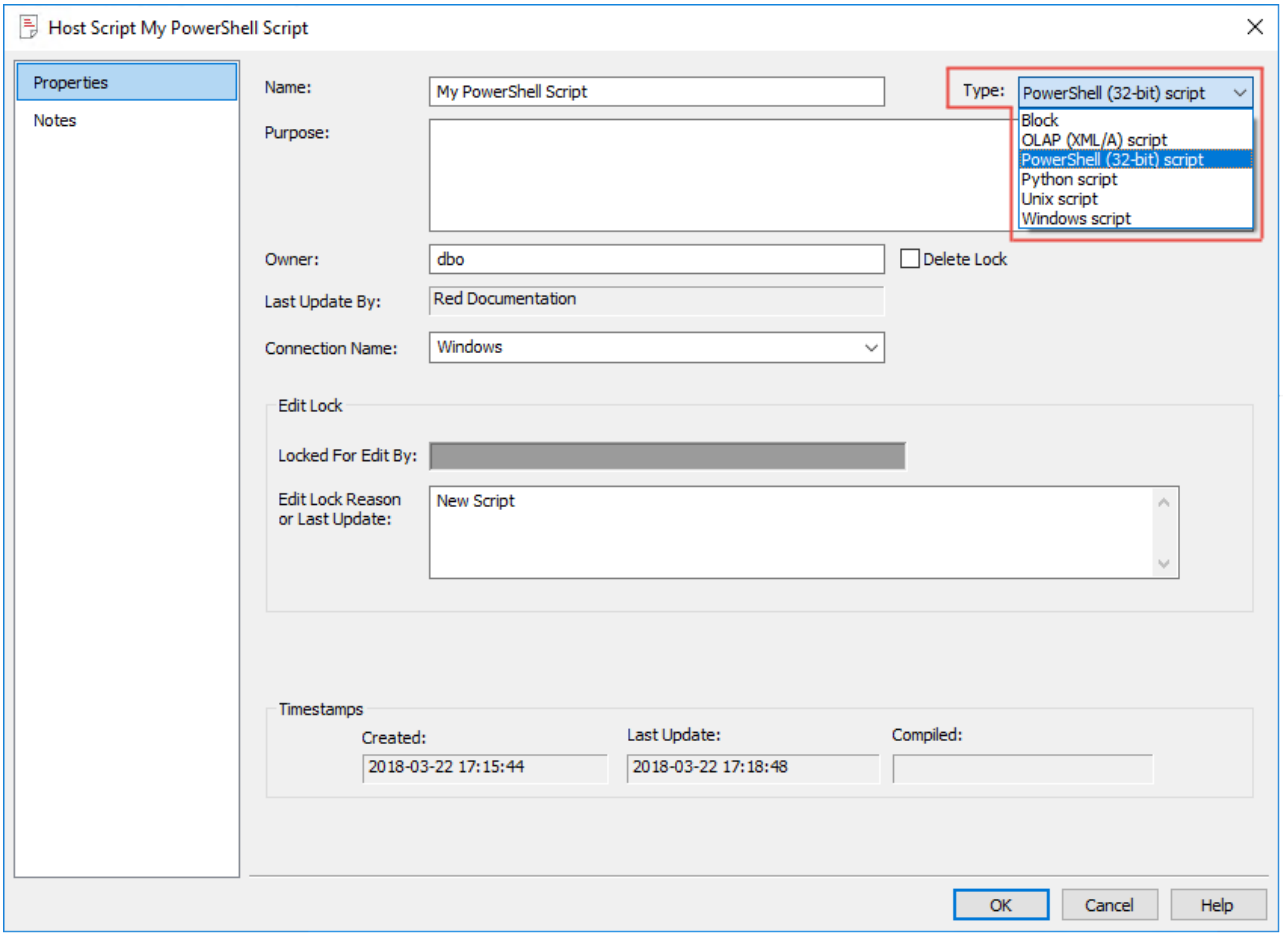


© Copyright, WhereScape Software Ltd All rights reserved.

Host Script Properties

The **Type** field specifies what type of Script object is being created in RED.

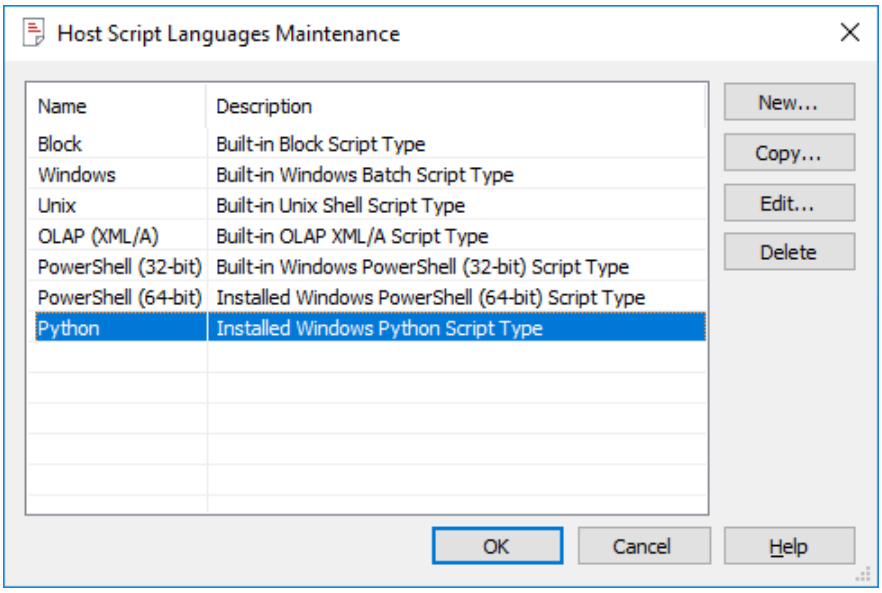
Refer to **Script Generation** for details.



© Copyright, WhereScape Software Ltd All rights reserved.

Maintaining Host Script Languages

The **Maintain Host Script Languages** screen displays a list of host script language types configured in RED and enables you to add a **New** host script language type. Clicking a host script language type from the list enables you to **Copy**, **Edit** or **Delete** the selected language type.



The following describe the elements of the **Maintain Host Script Languages** screen.

Host Script Languages

This pane lists the built-in and user defined host script language types that have been setup in RED. It details the name and description of each host script language available in RED.

Function Buttons

These buttons enable you to create and manage the user defined host script language type definitions.

Button	Description
New	Launches the **Edit Host Script Language** screen which enables you to create and specify the properties of the new host script language type. Refer to **Adding a Host Script Language Type** for details.
Copy	Duplicates the selected host script language type. All fields are populated with the values of the original host script language type which can be edited as required.
Edit	Launches the **Edit Host Script Language** screen which enables you to edit the selected host script language type.
Delete	Deletes the selected host script language type, if it is not currently in use. The user is notified if it is in use and cannot be deleted.

Notes:

- Built-in host script languages in RED can only be viewed, they cannot be edited or deleted. Only **user defined** host script language types can be edited or deleted.
- Host script languages that are currently in use cannot be deleted, RED displays a message if you attempt to delete a language type that is in use.

© Copyright, WhereScape Software Ltd All rights reserved.

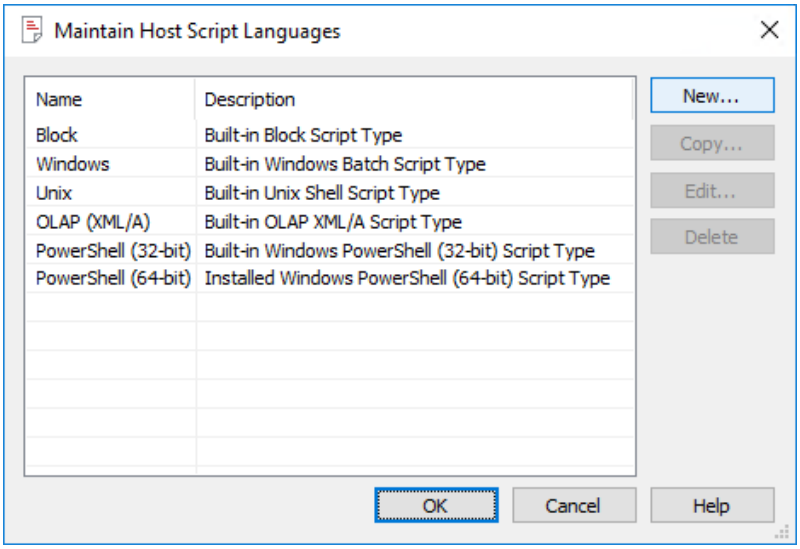
Adding a Host Script Language Type

To add a new host script language type, select **Tools > Host Script Languages > Maintain Host Script Languages**.

Notes:

For Unix host scripts for Load objects, RED leverages the Unix host script standard of defining the interpreter directive, after the Shebang (#!) character sequence in the first line of the Unix host script. This line provides the path to the command interpreter the operating system must use to execute the script. For example:
#!/usr/bin/perl -T
This enables RED users to also leverage a number of Unix host script languages for loading data into their Load objects.

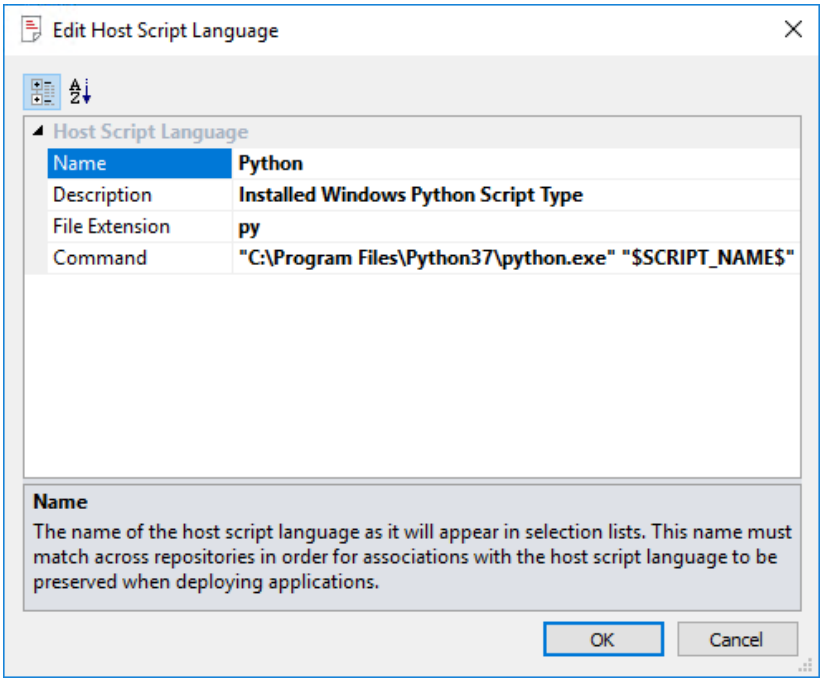
1. On the **Maintain Host Script Languages** screen, click **New**.



2. On the **Edit Host Script Language** screen, enter the properties of the host script language being added.

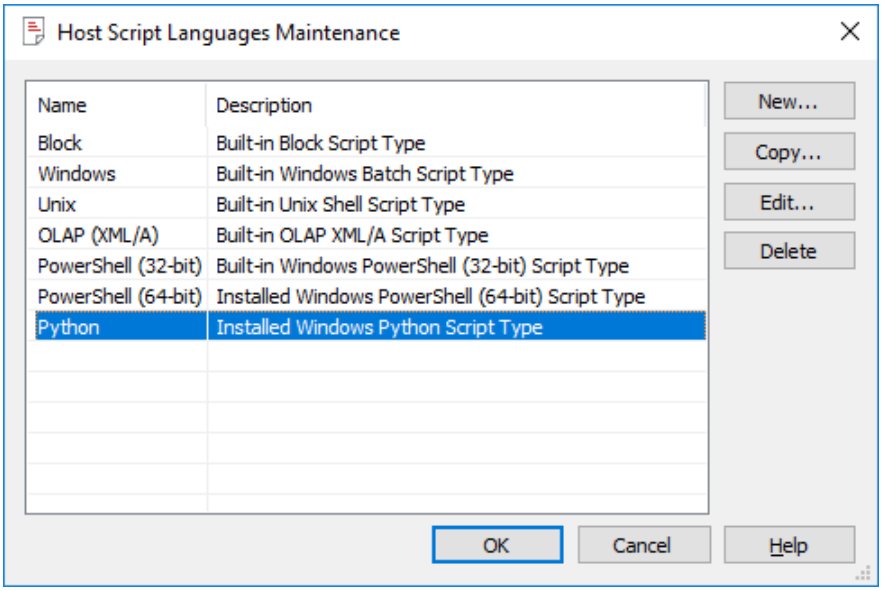
Tip:

Clicking a field name displays a description of the property at the bottom of the screen,



© Copyright, WhereScape Software Ltd All rights reserved.

3. Click **OK** to exit the **Edit Host Script Language** screen. The newly added host script language type is listed in the **Maintain Host Script Languages** screen.



4. Click **OK** to save the new host script language type.

Note:

User defined host script language definitions can be copied between RED meta data repositories, using **Tools > Import Metadata Objects** feature, which enables you to select an object (host script or template) to import that has the required host script type.

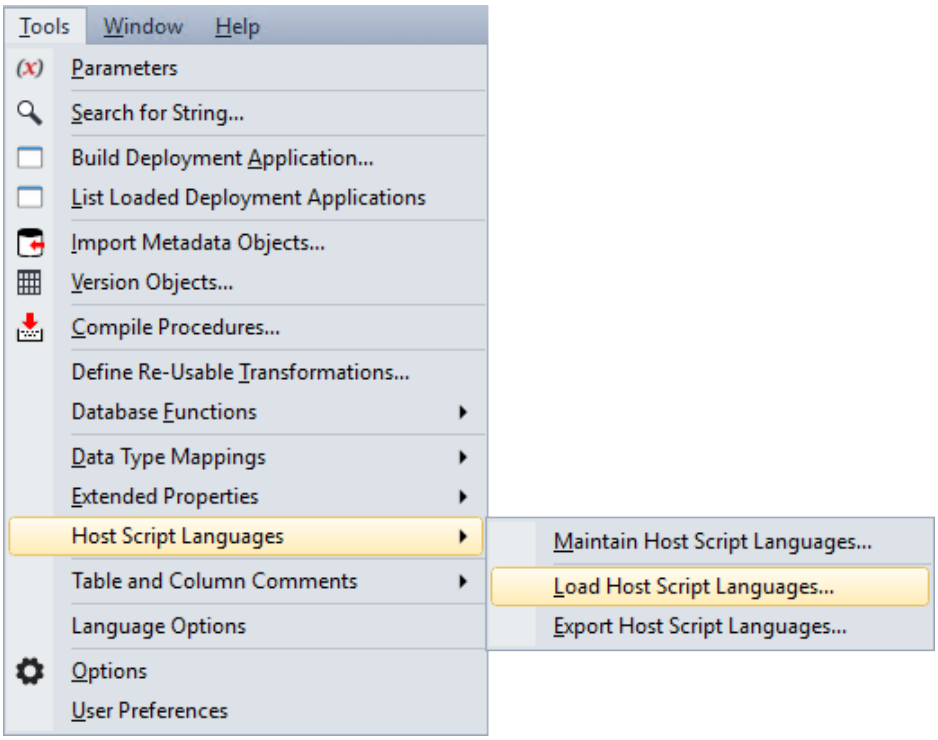
© Copyright, WhereScape Software Ltd All rights reserved.

Host Script Languages Data Migration Between Repositories

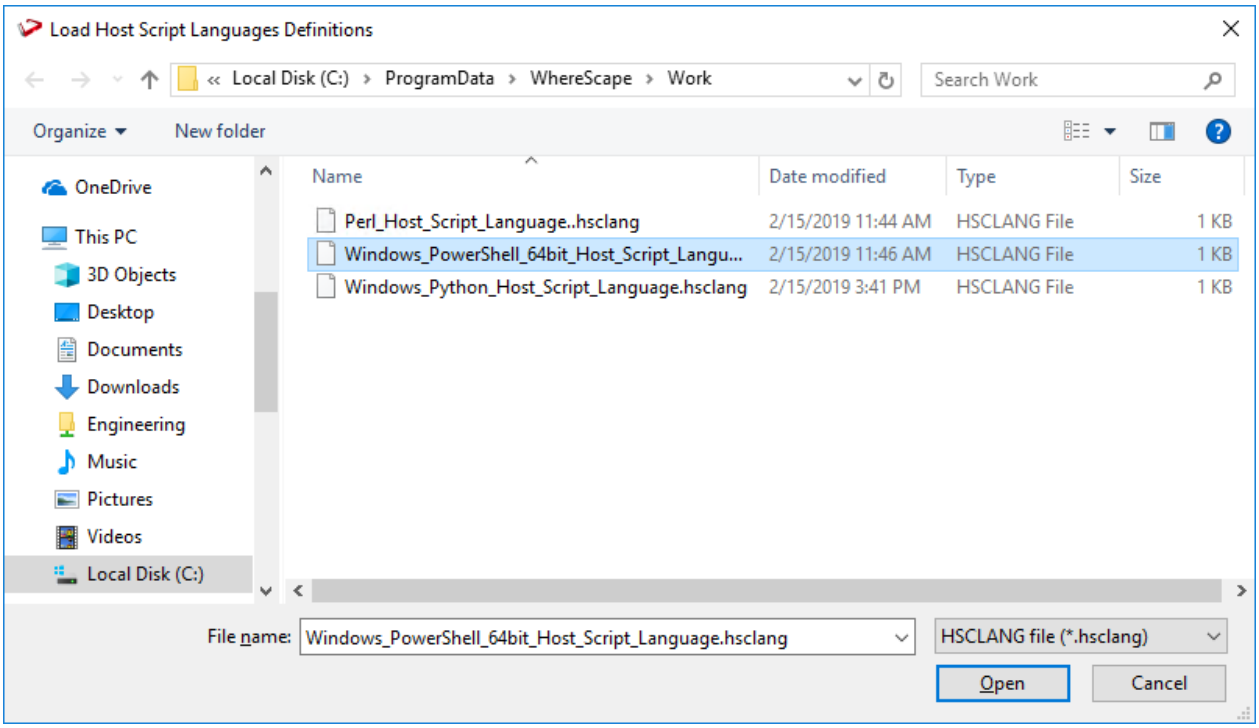
Host script languages defined and set can be propagated to the other RED repositories. The definitions are exported from the source repository and imported to the target repository via the **Tools > Host Script Languages** menu.

Loading Host Languages

To load host script language file, select **Tools > Host Script Languages > Load Host Script Languages**.

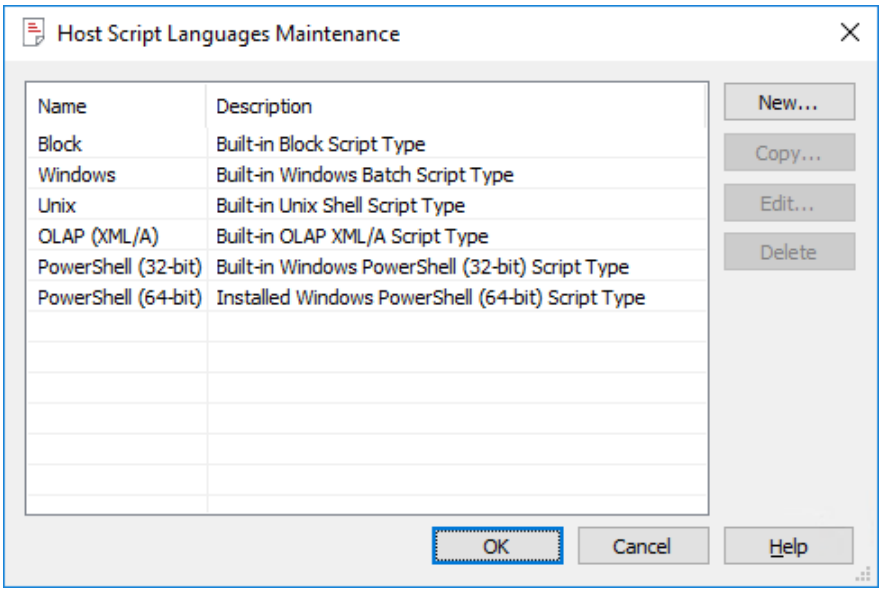


The following window is displayed. Select the host script language (HSCLANG) file to load . By default, RED expects the hsclang files to be in **Program Files\WhereScape\Work** directory, but this can be changed.

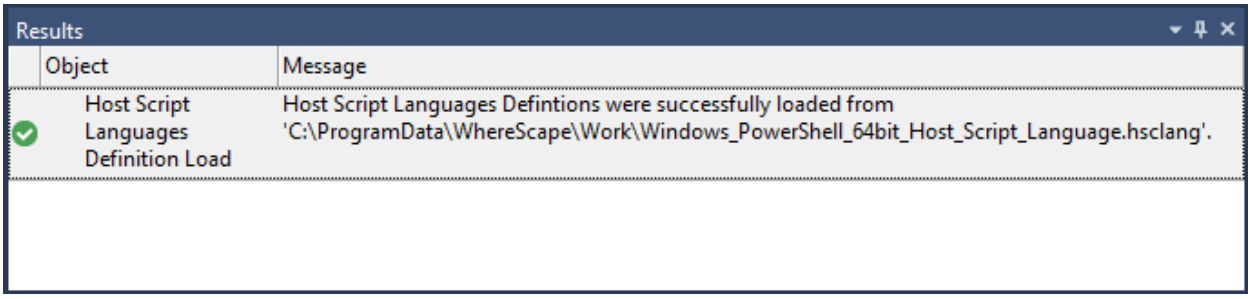


© Copyright, WhereScape Software Ltd All rights reserved.

The selected hsclang file is loaded and listed in the **Maintain Host Script Languages** screen.



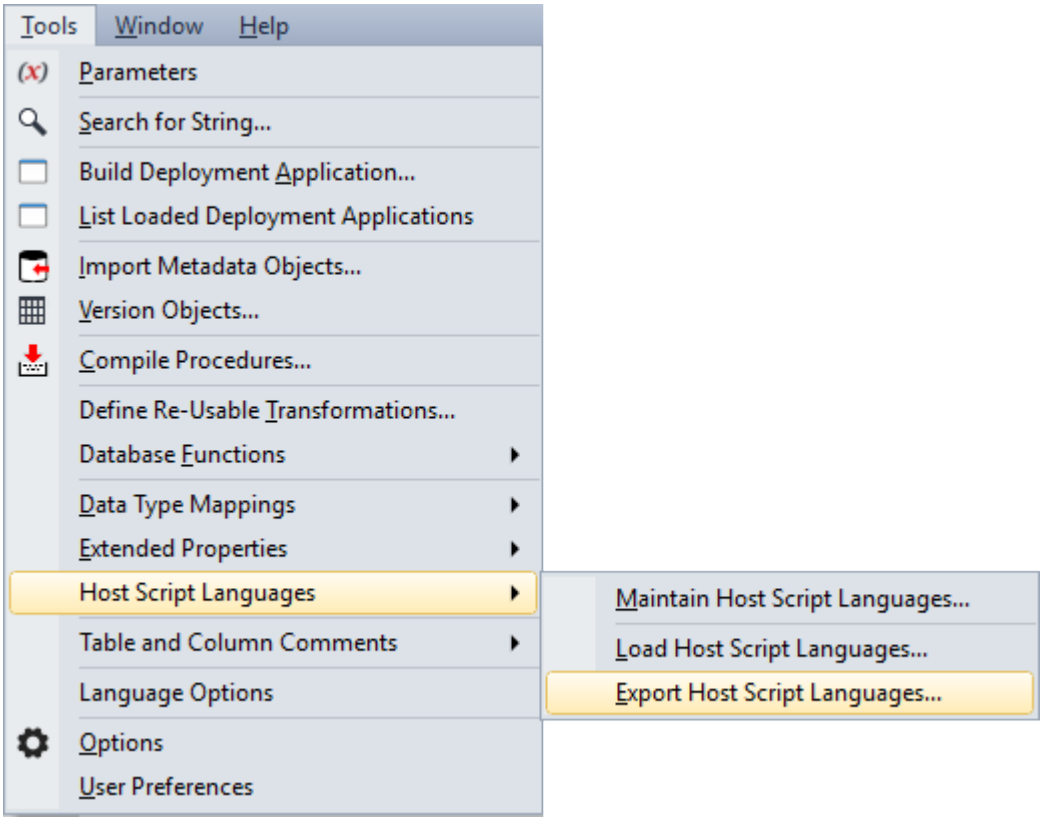
A confirmation message is displayed in the **Results** pane.



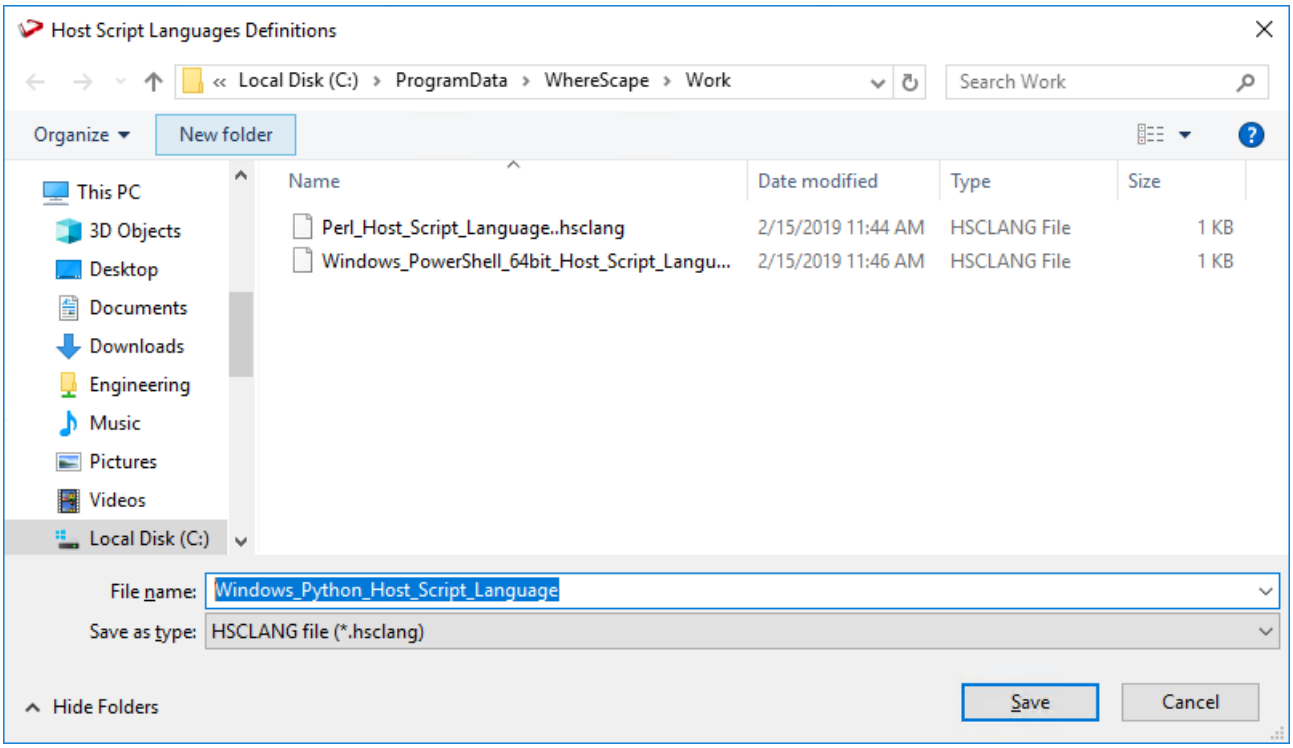
© Copyright, WhereScape Software Ltd All rights reserved.

Exporting Host Script Languages

To export host script languages from RED, select **Tools > Host Script Languages > Export Host Script Languages**.

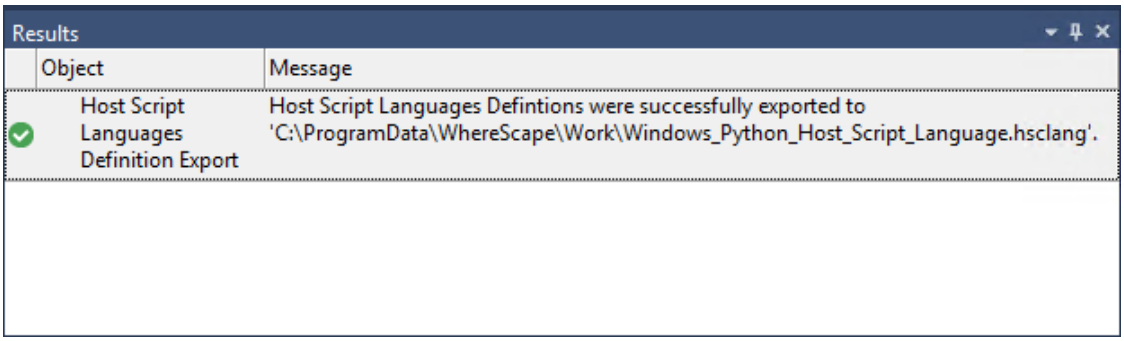


The following window is displayed. By default, RED exports the host script language (HSCLANG) file to **Program Files\WhereScape\Work** directory, but this can be changed. Type in the **File name** and then click **Save**.



© Copyright, WhereScape Software Ltd All rights reserved.

A confirmation message is displayed in the **Results** pane.



© Copyright, WhereScape Software Ltd All rights reserved.

Dedicated Command Line Interface (REDCLI)

WhereScape RED provides a dedicated command line interface called **REDCLI** which enables you to perform command functions. The **REDCLI** tool is available in the RED Windows installation directory. The tool can be called from 3rd party applications or Windows scripting languages to perform common functions.

REDCLI provides three usage modes:

Mode	Description
Standard	In this mode, one action is run and you authenticate to the RED repository for each action. The use case for this mode is if you only have one action to perform, e.g. deploying an application.
Interactive	In this mode, you connect to the target RED repository once and have a persistent connection to run multiple **REDCLI** actions. If you have several commands to run, using this mode removes the overhead of several database connections and reduces the command line options required for each action by not requiring the meta connection and logging options that were established on the initial interactive command line. This mode might be useful in cases where you have multiple actions to perform but have not prepared a batch file.
Batch	In this mode, you have a file containing a series of **REDCLI** actions that are executed sequentially. One use case is loading an enablement pack of templates, data type mapping sets, database function sets. This is particularly useful if you want to repeat a set of actions in multiple RED repositories.

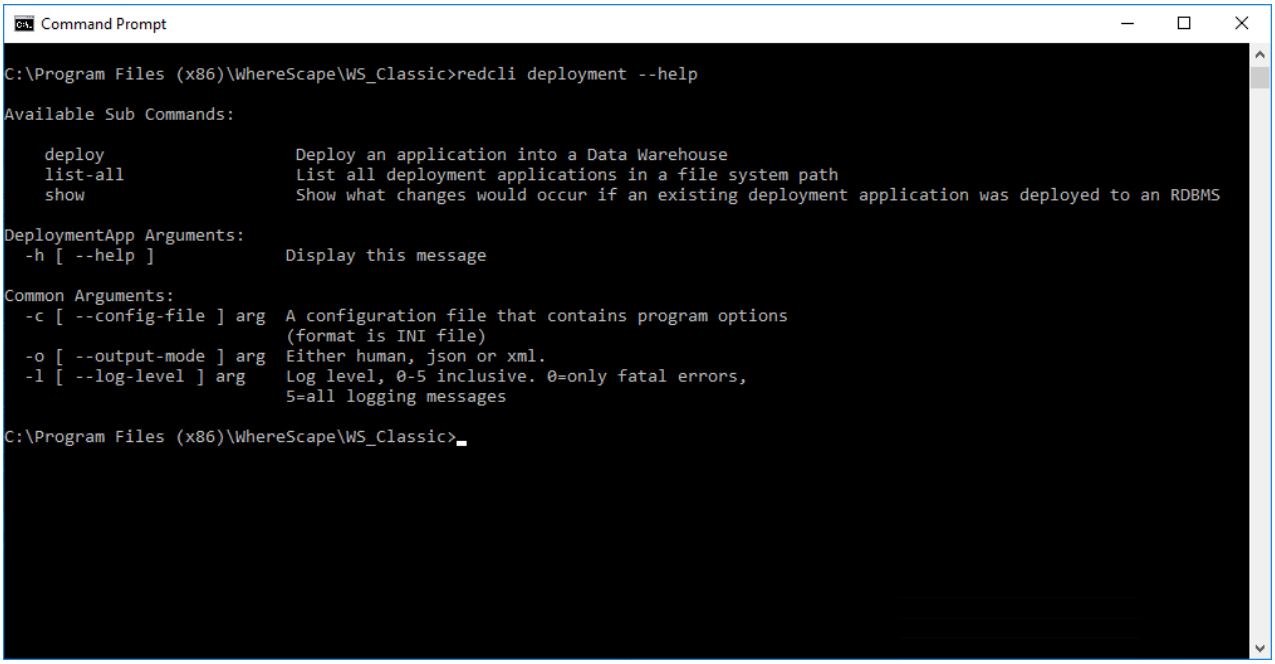
A command line help for each command within the **REDCLI** is available and the option level actions are as follows:

Command	Description
batch	Execute a batch of commands, and generate an empty batch file.
connection	Manage all aspects of connections.
dfs	Manage all aspects of database function sets.
dtm	Manage all aspects of data type mapping sets.
deployment	List, deploy and show the contents of deployment applications.
documentation	Create documentation for a metadata repository.
ext-prop-definition	Manage all aspects of extended property definitions.
ext-prop-value	Manage all aspects of extended property values.
script-lang-definition	Manage all aspects of host script language definitions.
interactive	Collect commands interactively from standard input and execute them.
license	Install, uninstall or show a license.
object	List information and generate update routines for objects in a metadata repository. The **REDCLI** command 'object generate-routine' works in conjunction with the **Generate update routine for objects** setting in the **Load Application > Metadata Options** window of the RED Setup Administrator. Refer to the section **Application Load Process** of the RED Install Guide for details. The default template for the table objects must be set in the target connection, before generating update routines via **REDCLI**. Refer to **Connection Routine Templates** for details.

© Copyright, WhereScape Software Ltd All rights reserved.

Command	Description
	Notes: Template-based update routine code generation for table objects: • Existing RED table objects uses the Template associated with the object, e.g. the current procedure or script that was used to generate the update routine. Refer to **Rebuilding Update Procedures > Generating an Update Procedure via a Template** for details. • New RED table objects uses the default update routine Template, set in the target connection. Refer to **Connection Routine Templates** for details.
odbc	List and validate ODBC DSNs in the system.
options	List, find, import and export Tools Options.
parameter	Manage all aspects of parameters.
powershell	Install WhereScape specific PowerShell modules.
procedure	Import, export and list all procedures.
repository	Create, upgrade and validate a metadata repository.
scheduler	Manage all aspects of scheduler configurations.
script	Import, export and list all scripts.
target	Manage all aspects of targets.
template	Import, export and list all templates.

Each command has its own help, for example `'RedCli deployment --help'` lists all available commands relevant to deployment applications.



Example to deploy a RED application:

```
REDCLI deployment deploy --app-number myproject --app-version v01 --app-directory
C:\Workspace\RedCli\Applications\ --options-location defaults --dest-connection-
name DataWarehouse --dest-target-name Schema --dest-scheduler-name qa_scheduler -
-continue-ver-mismatch --meta-database orcl12_ODW --meta-dsn ORCL-orcl12_ODW --
meta-password Wsl12345 --meta-user-name orcl12_ODW --output-mode json --log-level
5
```

© Copyright, WhereScape Software Ltd All rights reserved.

RED Command Line Interface

WhereScape RED provides command line interface applications that you can use to perform different tasks or execute functions and operations outside the RED GUI.

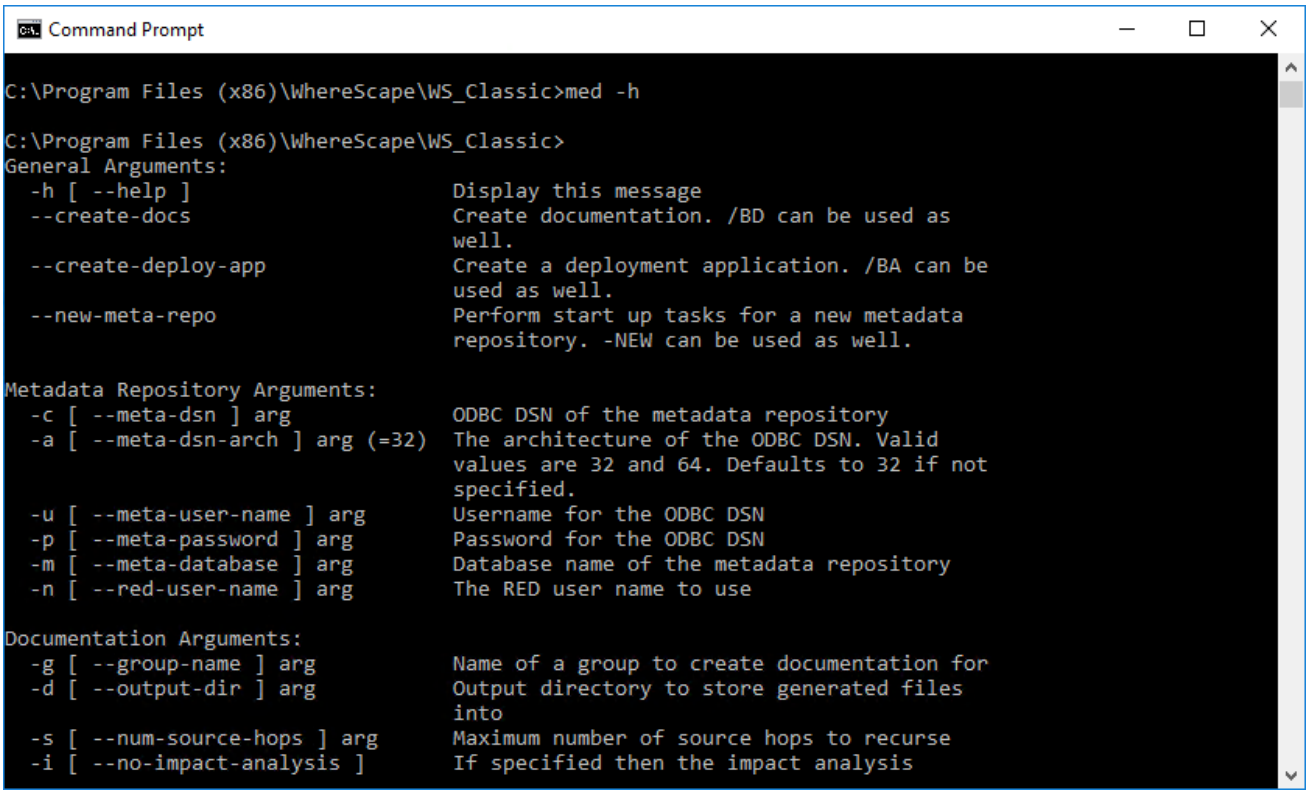
- RED Client Command Line
- RED Setup Administrator Command Line

RED Client Command Line (med.exe)

The RED Client command line interface enables you to execute commands to:

- Generate documentation in batch
- Create and deploy applications in batch
- Perform start up tasks for a new metadata repository

The following is a sample Windows Command Prompt that displays the help for RED Client (**med.exe**) arguments.



RED Client Command Line Arguments

The following describe the command line arguments that can be used in RED Client (**med.exe**) arguments.

Description	Argument long name	Argument short name
Display the RED arguments. Note: No other arguments required.	--help	-h or /h
Batch documentation create. Note: Must have other arguments, refer to	--create-docs	-BD or /BD

© Copyright, WhereScape Software Ltd All rights reserved.

Description	Argument long name	Argument short name
Batch Documentation Creation.		
Batch deployment application create. Note: Must have other arguments, refer to **Batch Deployment Application Creation**.	--create-deploy-app	-BA or /BA
First start with a new repository. Note: Must have other arguments, refer to **New Metadata Repository Start up Task Creation**.	--new-meta-repo	-NEW or /NEW
Group name Note: All objects included, if group name is not specified.	--group-name <name>	-G or /G
Multiple use: Output directory or Database schema.	--output-dir <path name> or <db_schema>	-D or /D
Number of processes/ hops in the source diagrams.	--num-source-hops <number>	-S or /S
Disable impact analysis	--no-impact-analysis	-I- or /I-
ODBC DSN	--meta-dsn <name>	-O or /O
ODBC user name	--meta-user-name <name>	-L or /L
ODBC DSN architecture	--meta-dsn-arch <32|64>	-A or /A
ODBC username	--meta-user-name <name>	-U or /U
ODBC password	--meta-password <password>	-P or /P
Connection name	--meta-dsn <name>	-C or /C
Metadata database Note: Only used in Teradata	--meta-database <name>	-M or /M
Session user name	--red-user-name <name>	-N or /N
Application ID number	--app-number <name>	-I or /I
Application version number	--app-version <number>	-V or /V
Application ID file name	--app-file-name <path name>	-AF or /AF
Remove connections	--remove-connections	-RC or /RC
Remove parameters	--remove-parameters	-RP or /RP
Remove scheduler jobs	--remove-jobs	-RJ or /RJ
Group name	--group-name <name>	-AG or /AG
Project name	--project-name <name>	-AP or /AP
All objects	--deploy-all	-ALL or /ALL

The succeeding sections outline how the arguments are used.

© Copyright, WhereScape Software Ltd All rights reserved.

Batch Documentation Creation

The following describe the relevant command line arguments for creating batch documentation (--create-docs or -BD):

Description	Argument long name	Argument short name
ODBC DSN	--meta-dsn <name>	-C or /C
ODBC DSN architecture	--meta-dsn-arch <32|64>	-A or /A
ODBC user name	--meta-user-name <name>	-U or /U
ODBC password	--meta-password <password>	-P or /P
Metadata database Note: Only used in Teradata	--meta-database <name>	-M or /M
RED user name	--red-user-name <name>	-N or /N
Group name Note: All objects included if group name is not specified.	--group-name <name>	-G or /G
Output directory	--output-dir <path name>	-D or /D
Number of processes/ hops in the source diagrams	--num-source-hops <number>	-S or /S
Disable impact analysis	--no-impact-analysis	-I- or /I-

Note:

Refer to **Batch Documentation Creation** for details.

Batch Deployment Application Creation

The following describe the relevant command line arguments for creating batch deployment application (--create-deploy-app or -BA):

Description	Argument long name	Argument short name
Output directory	--output-dir <path name>	-D or /D
ODBC DSN	--meta-dsn <name>	-C or /C
ODBC DSN architecture	--meta-dsn-arch <32|64>	-A or /A
ODBC user name	--meta-user-name <name>	-U or /U
ODBC password	--meta-password <password>	-P or /P
Metadata database Note: Only used in Teradata	--meta-database <name>	-M or /M
RED user name	--red-user-name <name>	-N or /N
Application ID number	--app-number <name>	-I or /I
Application version number	--app-version <number>	-V or /V
Application ID file name (e.g. app_id_xxx.wst)	--app-file-name <path name>	-AF or /AF
Remove connections	--remove-connections	-RC or /RC
Remove parameters	--remove-parameters	-RP or /RP
Remove scheduler jobs	--remove-jobs	-RJ or /RJ
Group name	--group-name <name>	-AG or /AG
Project name	--project-name <name>	-AP or /AP
All objects	--deploy-all	-ALL or /ALL

© Copyright, WhereScape Software Ltd All rights reserved.

Note:

Refer to **Creating and Loading Applications from the Command Line** for details.

New Metadata Repository Start up Task Creation

The following describe the relevant command line arguments for creating start up tasks for a new metadata repository (--new-meta-repo or -NEW):

Description	Argument long name	Argument short name
ODBC DSN	--meta-dsn <name>	-O or /O
ODBC DSN architecture	--meta-dsn-arch <32|64>	-A or /A
ODBC user name	--meta-user-name <name>	-L or /L
Metadata database	--meta-database <name>	-D or /D

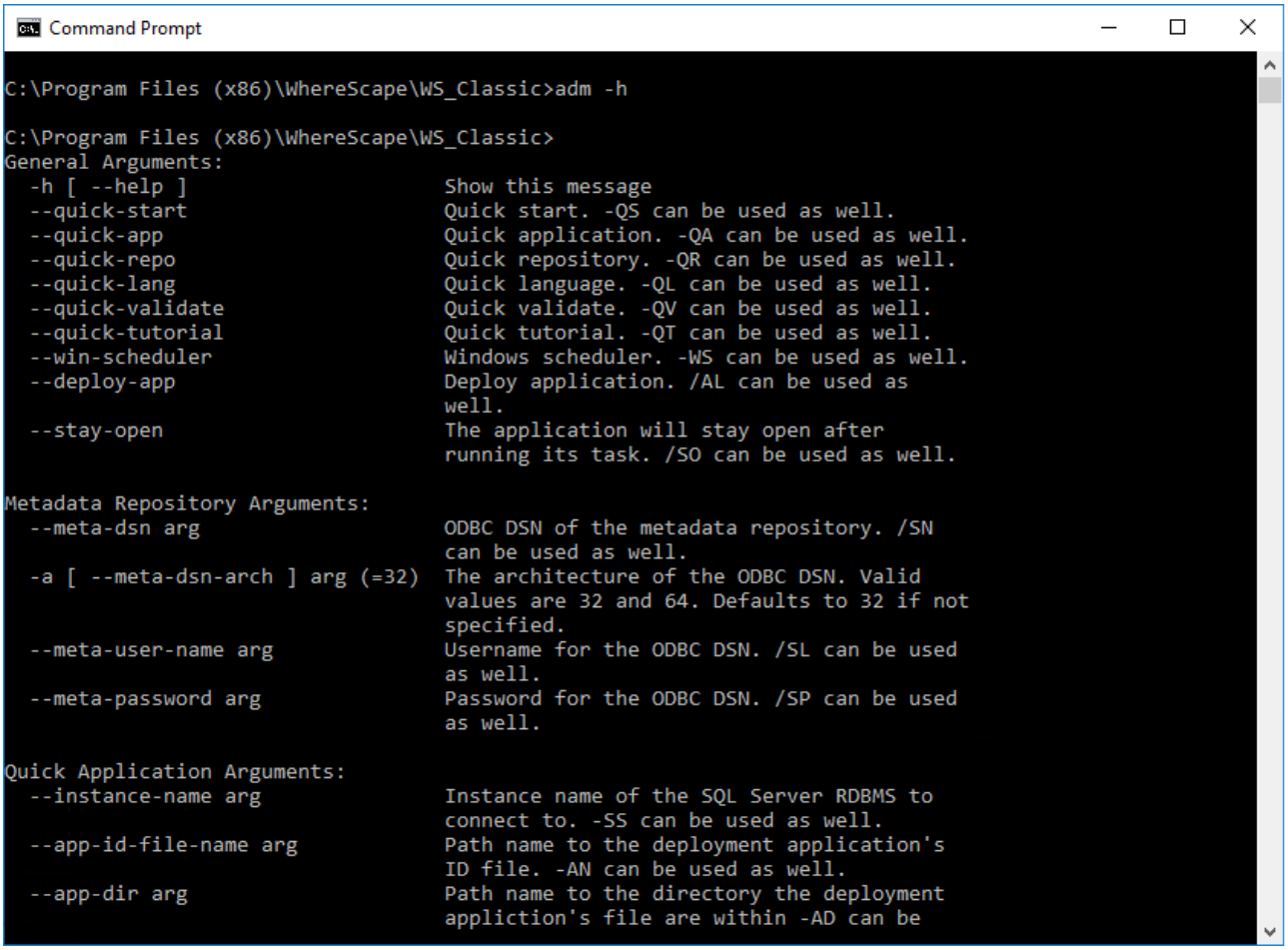
© Copyright, WhereScape Software Ltd All rights reserved.

RED Setup Administrator Command Line (adm.exe)

The RED Setup Administrator command line interface enables you to execute commands to :

- Quickly deploy a WhereScape RED application to create and populate a data warehouse
- Quickly create a RED repository with the required metadata tables
- Quickly load a file that contains language data into RED
- Quickly validate a RED repository and its metadata tables
- Quickly connect to a tutorial database to use in RED
- Install and configure a Windows scheduler for RED
- Deploy a RED application

The following is a sample Windows Command Prompt that displays the help for RED Setup Administrator (**adm.exe**) arguments.



RED Setup Administrator Command Line Arguments

The following describe the command line arguments that can be used in RED Setup Administrator (**adm.exe**).

Description	Argument long name	Argument short name
Display the RED arguments. Note: No other arguments required.	--help	-h or /h
Quick start Note: No other arguments required.	--quick-start	-QS or /QS
Quick application	--quick-app	-QA or /QA

© Copyright, WhereScape Software Ltd All rights reserved.

Description	Argument long name	Argument short name
Note: Must have other arguments, refer to **Quick Application**.		
Quick repository Note: Must have other arguments, refer to **Quick Repository**.	--quick-repo	-QR or /QR
Quick language Note: Must have other arguments, refer to **Quick Language**.	--quick-lang	-QL or /QL
Quick validate Note: Must have other arguments, refer to **Quick Validate**.	--quick-validate	-QV or /QV
Quick tutorial Note: Must have other arguments, refer to **Quick Tutorial**.	--quick-tutorial	-QT or /QT
Windows scheduler Note: Must have other arguments, refer to **Windows Scheduler**.	--win-scheduler	-WS or /WS
Deploy application Note: Must have other arguments, refer to **Deploy Application**.	--deploy-app	-AL or /AL
Stay open Notes: • Use to make Setup Administrator remain open, after the application has been created. • Relevant for -AL LOAD option. • Relevant for -QA, -QR, -WS, -QL and -QV application switch options.	--stay-open	-SO or /SO

Note:

When --quick-start (-QS or /QS) is specified, then no other parameters are required.

The succeeding sections outline how the arguments are used.

© Copyright, WhereScape Software Ltd All rights reserved.

Quick Application

The following describe the relevant command line arguments for quick application (--quick-app or -QA):

Description	Argument long name	Argument short name
Instance name	--instance-name <name>	-SS <name> or /SS <name>
New repository name	--meta-dsn <name>	-SN <name> or /SN <name>
Database login name	--meta-user-name <name>	-SL <name> or /SL <name>
Database password	--meta-password <password>	-SP <password> or /SP <password>
Application ID file name	--app-id-file-name <name>	-AN <file name> or /AN <name>
Application file's directory	--app-dir <path name>	-AD <directory name> or /AD <name>
Application load instance	--app-instance <name>	-LI <name> or /LI <name>
Application trusted flag	--app-trusted	-LT or /LT
Application ODBC DSN	--app-meta-dsn <name>	-LD <name> or /LD <name>
ODBC DSN architecture	--meta-dsn-arch <32|64>	-a <32|64> or /a <32|64>
Application load database	--app-db-name <name>	-LN <name> or /LN <name>
Application job name	--app-job-name <name>	-JN <name> or /JN <name>
Datawarehouse connection name	--dw-con-name <name>	-DC <name> or /DC <name>
MSAS cube connection name	--msas-con-name <name>	-CC <name> or /CC <name>
MSAS cube database name	--msas-db-name <name>	-CD <name> or /CD <name>
Scheduler login name	--sched-login-name <name>	-ML <name> or /ML <name>
Scheduler password	--sched-password <password>	-MP <password> or /MP <password>

Note:

Refer to the **SQL Server Quick Application** section of the WhereScape RED Setup Administrator User Guide for details.

Quick Repository

The following describe the relevant command line arguments for quick repository (--quick-repo or -QR):

Description	Argument long name	Argument short name
ODBC DSN	--meta-dsn <name>	-SN <name> or /SN <name>
ODBC DSN architecture	--meta-dsn-arch <32|64>	-a <32|64> or /a <32|64>
ODBC login name	--meta-user-name <name>	-SL <name> or /SL <name>
ODBC password	--meta-password <password>	-SP <password> or /SP <password>
TDPID	--tdpid <id>	-DI <name> or /DI <name>
Dimension tablespace	--dim-space <name>	-DS <name> or /DS <name>
Index tablespace	--index-space <name>	-IS <name> or /IS <name>
Metadata tablespace	--meta-space <name>	-MS <name> or /MS <name>

Note:

Refer to the **SQL Server Quick Application** section of the WhereScape RED Setup Administrator User Guide for details.

© Copyright, WhereScape Software Ltd All rights reserved.

Quick Language

The following describe the relevant command line arguments for quick language (--quick-lang or -QL):

Description	Argument long name	Argument short name
ODBC DSN	--meta-dsn <name>	-SN <name> or /SN <name>
ODBC DSN architecture	--meta-dsn-arch <32|64>	-a <32|64> or /a <32|64>
ODBC login name	--meta-user-name <name>	-SL <name> or /SL <name>
ODBC password	--meta-password <password>	-SP <password> or /SP <password>
Language file	--lang-file-name <name>	-LF <path name> or /LF <path name>
Add language if absent	--add-lang	-Add or /ADD

Note:

Refer to the **Languages** section of the WhereScape RED Setup Administrator User Guide for details.

Quick Validate

The following describe the relevant command line arguments for quick validate (--quick-validate or -QV):

Description	Argument long name	Argument short name
ODBC DSN	--meta-dsn <name>	-SN <name> or /SN <name>
ODBC DSN architecture	--meta-dsn-arch <32|64>	-a <32|64> or /a <32|64>
ODBC login name	--meta-user-name <name>	-SL <name> or /SL <name>
ODBC password	--meta-password <password>	-SP <password> or /SP <password>
Dimension tablespace	--dim-space <name>	-DS <name> or /DS <name>
Index tablespace	--index-space <name>	-IS <name> or /IS <name>
Metadata tablespace	--meta-space <name>	-MS <name> or /MS <name>

Note:

Refer to the **Validating a Metadata Repository using the Command Line** section of the WhereScape RED Setup Administrator User Guide for details.

Quick Tutorial

The following describe the relevant command line arguments for quick tutorial (--quick-tutorial or -QT):

Description	Argument long name	Argument short name
ODBC DSN	--meta-dsn <name>	-SN <name> or /SN <name>
ODBC DSN architecture	--meta-dsn-arch <32|64>	-a <32|64> or /a <32|64>
ODBC login name	--meta-user-name <name>	-SL <name> or /SL <name>
ODBC password	--meta-password <password>	-SP <password> or /SP <password>

Note:

Refer to the **Tutorial Data** section of the WhereScape RED Install Guide for details.

© Copyright, WhereScape Software Ltd All rights reserved.

Windows Scheduler

The following describe the relevant command line arguments for Windows scheduler (--win-scheduler or -WS):

Description	Argument long name	Argument short name
Scheduler name	--sched-name <name>	-ID <name> or /ID <name>
ODBC DSN	--meta-dsn <name>	-SN <name> or /SN <name>
ODBC DSN architecture	--meta-dsn-arch <32|64>	-a <32|64> or /a <32|64>
ODBC login name	--meta-user-name <name>	-SL <name> or /SL <name>
ODBC password	--meta-password <password>	-SP <password> or /SP <password>
Metadata schema/database	--meta-database <name>	-MS <name> or /MS <name>
Service login name	--service-user-name <name>	-LN <name> or /LN <name>
Internal scheduler name	--int-sched-name <name>	-IN <name> or /IN <name>
Log level	--log-level <number>	-LL <number> or /LL <name>
Polling interval	--poll-interval <number>	-PI <number> or /PI <name>
Work directory	--work-dir <path name>	-WD <path name> or /WD <path name>
Log file name	--log-file-name <name>	-WL <path name> or /WL <path name>
TNS name	--tns-name <name>	-TN <name> or /TN <name>
Service password	--service-password <password>	-LP <password> or /LP <password>
Scheduler load type Notes: • If omitted, the scheduler load type defaults to ODBC. • The legacy option DL can be used to set only 32 bit OCI load type.	--sched-load-type <ODBC|OracleOCI|OracleOCI64>	-DL or /DL is equivalent to --sched-load-type "OracleOCI"

Note:

Refer to the **Installing a Windows Scheduler** section of the WhereScape RED Install Guide for details.

Deploy Application

The following describe the relevant command line arguments for deploy application (--deploy-app or -AL):

Description	Argument long name	Argument short name
Application ID file name	--app-id-file-name <path name>	-AF <file name> or /AF <name>
Application parameter file name	--app-param-file-name <path name>	-PF <file name> or /PF <name>
Log file name	--log-file-name <path name>	-LF <path name> or /LF <path name>

Note:

Refer to the **Creating and Loading Applications from the Command Line** for details.

© Copyright, WhereScape Software Ltd All rights reserved.